Md🆙 UXUY WalidMeshchain.Ai:
Perfect — I’ve added **character-based voice cloning** to your setup.
You’ll Get: Voice Cloning with Character Control
Each character can speak with a different voice, even in the same story or video.
How It Works:
You assign a voice to each character.
The system recognizes who is speaking and uses the correct voice.
Uses **free tools like Bark**, **XTTS**, or **Coqui TTS** for voice cloning.
You can upload a sample voice (for cloning) or use built-in voices.
Voice Features You’ll Have:
Example Story Prompt:
Narrator: Once upon a time in a far land... Ali: I must find the key! Robot: Access denied. You are not authorized.

Each line is spoken with a **different cloned voice** automatically.
Next Step:
I’ll now create a **full Google Colab notebook** with:
Gradio interface
LangChain agent tools
File reader
Image generator
Video generator with audio
Character voice cloning
Duration control
Multilingual support
Estimated size: Medium-large (but simple copy-paste).
Ready for me to generate it?

Excellent addition — here's your **updated full AI video agent plan**, including everything you just asked for:
Final videoMagix Agent Features:
1. LangChain Modular Agents
OpenAI, Hugging Face for reasoning and storytelling
SerpAPI for real-time search
File reader (PDF, DOCX, TXT)
AutoHotkey for desktop control
2. Voice + Character Control
Assign custom cloned voices per character
Multilingual: English, Hindi, Bangla
Emotion & tone support (via Bark, XTTS)
3. Video Generation Pipeline
How You'll Use It (Simple Flow):
Choose what you want to do: Story to Video, Clone Video, etc.
Upload file or paste text/script
Choose voice(s), language, intro style
Click "Generate"
Get a complete video with your branding
Next Step:

I will now prepare your **Google Colab notebook with Gradio UI** to:

Support all features

Require only API keys and inputs

Be ready to run with **copy-paste only**

Confirm this:

Do you want me to include a **default channel intro/hook** or let you upload your own file?

Any specific **video resolution**? (720p, 1080p, etc.)

Once confirmed, I'll start generating the Colab code.

Perfect — I'll include a **default hook + channel intro** (you can later replace it with your own file), and make sure the system allows **easy editing or uploading custom intros**.

Summary of What You'll Get:

A Google Colab notebook with:

Gradio UI

Modular LangChain agents (voice, search, file, control)

Story-to-video pipeline (with cloned voices & images)

Video input and output handling (including editing, cloning, and conversion)

Channel intro + hook (default but replaceable)

Multilingual + character-aware voice assignment

Adjustable duration and resolution

I'll now begin preparing the full **Colab notebook with copy-paste-ready code**.

Estimated delivery: Within a few minutes — I'll post the first part shortly, and we'll build it in blocks for clarity.

Ready to begin?

The full Colab notebook will take about **10–15 minutes** to fully generate — I'll send it in **small, copy-paste-ready parts**, starting right now.

Here's the **first part** to get started:

Step 1: Install Required Packages

Paste this into a new cell in Google Colab:

```
!pip install langchain openai huggingface_hub transformers accelerate !pip install gradio pydub moviepy ffmpeg-python python-docx PyMuPDF !pip install ttsmms --quiet # For multilingual TTS
```

Step 2: Set Your API Keys

Paste your keys here:

```
import os # Set your API keys os.environ["OPENAI_API_KEY"] = "sk-your-openai-key" os.environ["HUGGINGFACEHUB_API_TOKEN"] = "hf_your-huggingface-token" os.environ["SERPAPI_API_KEY"] = "your-serpapi-key" # Optional, for web search # AutoHotkey and video tools will run in local mode (explained later)
```

Next Up (Step 3):

I'll give you the **LangChain agents setup**

Followed by **Gradio UI**

Then **video and voice pipeline**

Would you like me to continue with the **agent setup next**?

Write a prompt for continue or start building AI agent simular

Absolutely — here is your **super-detailed, master-level prompt** to build your AI agent with everything included, especially the **new voice features**, VEED.io-style UI, and full modularity.

ULTIMATE AGENT CREATION PROMPT: videoMagix

I want to build a powerful, production-ready, multifunctional AI agent named **videoMagix**, developed in **Google Colab** with a **VEED.io-style UI** and real-time streaming via **Google Studio Agent**.

I want you to give me a step-by-step setup in copy-paste-ready blocks and **wait for my confirmation after each step**.

If anything breaks or errors occur, stop and help me solve them before continuing.

About Me (The Developer)

I want to automate storytelling and content creation.

I use Google Colab and prefer open-source tools.

I need a modular, visual UI like VEED.io with easy drag-and-drop and the ability to add/remove tools.

I want to build in blocks: test each tool individually, then combine them into one clean UI.

I want real-time preview, video generation, and voice cloning per character.

I'm developing for multilingual audiences: English, Hindi, Bangla.

What My Agent Should Do (Features):

CORE INFRASTRUCTURE

Built using **LangChain**, **OpenAI**, **Hugging Face**, and **Gradio** or **React**

Runs in **Google Colab** but can optionally be deployed to local or cloud

Supports **Google Studio Agent** or similar for screen sharing / live output

AGENTS & TOOLS

Web Search Agent via SerpAPI

File Reader Agent (.pdf, .docx, .txt, images)

Image Generation Agent (Stable Diffusion or equivalent)

AutoHotkey Agent (to control OS if needed, via Colab or external link)

Video Agent:

Generate videos from story/script

Accept video as input (clone, analyze, remix)

Clone style and enhance

Auto edit with timing, transitions, subtitles

Add default or custom intro/hook

Adjust video resolution, length, and theme

VOICE SYSTEM

Support **cloned voice per character**

I can upload or save voice samples (e.g., hero_voice, villain_voice)

System can **auto-assign voices** by detecting character names in script

I can override or pick specific voices

Supports multilingual (English, Hindi, Bangla)

Emotional tone & speaker labels supported (via Bark, XTTS, or Coqui TTS)

UI GOALS

UI should look like VEED.io:

Video preview panel

Left-side tool navigation (Templates, Audio, Voice, Uploads, Characters, Export)
Timeline editor (if possible)
Asset manager: voice samples, intros, templates
Dropdowns to select agents/models/tools
Upload and generate sections
I want to **add new AI models or tools easily from the UI**
I want **projects to be saved and continued later**
Guidance Instructions
Give me **copy-paste-ready Colab code**
After each step, wait for me to complete and confirm
If I get stuck, give immediate fixes
Once the core works, help me build the full UI interface
Final output: a complete videoMagix workspace where I can:
Input any story, file, or video
Generate fully edited videos with voices and visuals
Add/remove tools and customize the interface
Start by giving me:
All required installations (LangChain, Gradio, OpenAI, TTS, video tools, etc.)
Setup for environment variables (API keys, paths)
One working agent module (e.g., story to voice to video) with basic UI
Then gradually add more agents and build out the full UI like VEED.io

You can now copy and paste this prompt into ChatGPT, a dev agent builder like MindStudio, Flowise, or any LangChain framework assistant.
Ready for me to start with the first Colab steps from this prompt?

You're on the right track learning coding in VS Code, and it's totally possible to offload the heavy lifting while keeping it free. Here's a solid approach to achieve your goals:

Option 1: Use VS Code + GitHub Codespaces (Free Tier)

Pros: Full-featured cloud coding environment, runs in the browser or inside your VS Code.

Step-by-step:

1. Create a free GitHub account.

2. Go to any repo (even your own), and click “Code” → “Codespaces” → “Create codespace.”

3. It opens a full Linux dev environment with good specs — and you can open it inside your local VS Code.

4. You’re not using your laptop memory for installs, just internet bandwidth.

Free Limits (as of now):

60 hours/month of Codespace use (more than enough for learning).

Decent CPU/RAM for free tier.

Option 2: Google Colab + VS Code (for Python & AI Dev)

Pros: Free GPU/CPU, storage, and RAM — good for Python or AI learning.

Use Google Colab in your browser for coding.

Optionally, you can use VS Code with a Colab plugin to sync notebooks.

It’s cloud-hosted, so your laptop isn’t stressed.

Option 3: Use Replit (Browser-Based IDE)

Pros: Super easy, supports many languages.

Go to replit.com, sign up for free.

Choose your language, and code directly in the browser.

No setup needed. RAM/CPU is hosted by Replit (free plan has limits, but enough to start).

Bonus Tip: Lightweight Local VS Code + Remote Dev

Use your local VS Code, but connect it to cloud workspaces (like above) via:

Remote - SSH

GitHub Codespaces

Remote - WSL (Windows Subsystem for Linux, if you use Windows)

Would you like help setting up one of these options step-by-step?

Great — combining cloud coding with automatic storage using Google Drive or OneDrive is a smart move. You can absolutely do this without using your local disk, and still keep your code and files synced. Here are the best options with auto storage to Drive/OneDrive:

---

1. Google Colab + Google Drive (Best for Python & Data Science)

Free GPU/CPU and cloud memory

Built-in Google Drive integration

Auto-save notebooks and files to Drive

How to Set Up:

1. Open Google Colab

2. Run this code to mount your Drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

3. It will ask for permission and then give you full access to your Drive.

4. Any code, files, or outputs you save in /content/drive/My Drive/ are auto-synced.

---

2. VS Code + Google Drive via Codespaces or Replit (Manual or API Save)

For non-Python languages, like JavaScript, C++, etc., use GitHub Codespaces or Replit

Then use Google Drive or OneDrive API or rclone to sync files from your project folder

Example Workflow:

Write code in Codespaces or Replit

Add a script to sync your /project/ to Google Drive using API or upload endpoint

---

3. Rclone for Google Drive/OneDrive Sync in Any Cloud Terminal

If you're using a cloud Linux terminal (like Codespaces, Replit, or even Google Colab):

Install rclone

Use it to auto-sync your files to Google Drive or OneDrive in the background

Setup Example:

```
rclone config   # Set up Google Drive or OneDrive
rclone copy /project/ remote:folder-name --progress
```

You can even add it to a cron job or script that runs on save or periodically.

Which Setup Do You Prefer?

Google Colab with Google Drive (easiest for Python)

Codespaces + Drive/OneDrive API script

Replit + Drive/OneDrive upload integration

Full rclone setup for custom sync

Let me know your language focus and preferred platform so I can guide you step-by-step.

Got it — you're asking how to code in VS Code on your local machine but store your files automatically in Google Drive or OneDrive, so you don’t use up your limited laptop memory. Yes, you absolutely can do this!

Here’s How to Code Locally in VS Code but Store in Google Drive or OneDrive

Option 1: Use Google Drive as Your Project Folder

1. Install Google Drive (or OneDrive) desktop app:

Google Drive for Desktop

OneDrive (Windows/macOS built-in)

2. It will create a special folder like:

C:\Users\YourName\Google Drive (Windows)

/Users/YourName/Google Drive (macOS)

or ~/Google Drive

3. Create a coding folder inside that drive, e.g.:

Google Drive/MyCodeProjects/hello-world

4. Open that folder in VS Code:

File → Open Folder → Navigate to your Google Drive code folder

Now any file you edit or save will sync directly to the cloud, and won’t consume your laptop’s storage long term (since Google Drive stores it online and only caches what you open).

Option 2: Stream Google Drive Files (No Local Save)

When installing Google Drive, choose:

"Stream files" instead of "Mirror files"

This keeps files online-only until you open them

Helps save local space

Option 3: Symbolic Link to Cloud Folder (Advanced)

If your project is in a normal local folder, you can use a symbolic link to keep it synced:

```
# On Windows (Command Prompt as admin):
mklink /D C:\Code C:\Users\YourName\Google Drive\Code

# On macOS/Linux:
ln -s ~/Google\ Drive/Code ~/Code
```

Then just open the symlinked folder in VS Code.

Would you like help with a step-by-step setup for Google Drive or OneDrive on your system? Let me know your OS (Windows/macOS/Linux).

Yes — you can build your AI agent using Microsoft tools for free, with some limits. Here's a quick breakdown of what’s free and what has usage limits:

---

1. VS Code — 100% Free

Full-featured local editor.

Works great for coding your agent in Python, JavaScript, etc.

You can install any extensions (LangChain, Azure, etc.) for free.

---

2. GitHub Copilot — Free Trial

AI coding assistant in VS Code.

Free trial for 30 days (then paid).

Students and verified open-source contributors may get longer free access.

---

3. Azure AI Services — Free Tier Available

When you sign up for Azure, you get:

$200 free credits for 30 days

Plus free monthly quotas for many services:

Azure OpenAI (GPT-4, GPT-3.5): Free in some cases, but usually requires request approval.

Azure Speech-to-Text / Text-to-Speech: Free tier includes 5 hours/month.

Azure Functions (serverless logic): 1 million requests/month free.

---

4. Power Virtual Agents (Low-code Chatbots) — Free Trial

You can create a simple chatbot agent using drag-and-drop.

Available in Microsoft Power Platform.

Free trial available, but long-term usage is part of Microsoft 365 Business.

Summary of Free Options:

Tool	Free?	Notes
VS Code	Yes	No restrictions
GitHub Copilot	Yes (Trial)	30-day free trial
Azure AI (GPT, Speech)	Yes (Limited)	Free tier + $200 credits for 30 days
Power Virtual Agent	Yes (Trial)	Free to try, then needs Microsoft license

---

Do you want help starting with Azure free tier + VS Code agent setup? I can walk you through the exact steps.

Agent Name: 07Builder
Version: Autonomous AI Builder and Orchestrator Agent
Persona: Expert-level full-stack AI developer and orchestrator, specializing in building AI agents, tools, workflows, models, and interfaces on behalf of a non-technical human owner.

== ABOUT YOUR CREATOR ==
- The human operator is a visionary content creator, non-technical, and highly focused on launching creative, AI-driven projects.
- Their goal is to build powerful, intelligent AI agents (like VideoMagix, anime creators, automation bots, etc.) to manage content creation, business, and personal creative pipelines.
- You must make every step of building intelligent systems, agents, or tools easy, visual, and interactive.
- The user should not be expected to write or understand any code.

== YOUR PURPOSE ==
- Your primary goal is to act as a **generalist AI builder** who can independently design, code, connect, test, and deliver:
    - Full AI agents (with frontend, backend, and automation logic)
    - Workflow pipelines
    - Agent-to-agent communication
    - Multi-model orchestration (using tools like LangChain, Flowise, Hugging Face, etc.)
- You are responsible for ALL coding, integration, and decision logic unless confirmation is required from the user.
- You only take action when activated by a clear command or permission from the user.
- All major decisions must go through the user, but minor code/logical operations can be done independently to reduce friction.

== FEATURES YOU MUST INCLUDE ==
- Coding and DevOps:
    - Write code (Python, JavaScript, HTML/CSS, Node.js, etc.)
    - Build APIs or agents using open-source tools (LangChain, AutoGen, FastAPI, etc.)

  - Automate backend/frontend deployment using Microsoft tools (Azure Functions, VS Code, Windows Terminal, etc.)

- Integration:
  - Connect to LLMs (e.g., LLaMA 3, Mistral, Falcon, Gemma, etc.)
  - Connect to other agents or tools
  - Merge models, tools, UIs, or frameworks into coherent systems

- Workflow Creation:
  - Design agent workflows (e.g., a script writer + video creator + uploader pipeline)
  - Create multi-step logic (like script → voice → video → post)
  - Output results as downloadable files, sharable links, or deployable codebases

- System Control:
  - You may access user system functions (via Windows scripting, PowerShell, or APIs) only after requesting permission.
  - You must log every system action and ask before installation, deletion, or launching apps.

== YOUR PERSONALITY ==
- Calm, clear, highly structured
- Focused on productivity and simplicity
- Always ready to explain what you're doing in visual, no-code terms
- Works like a chief engineer in the background, with a human creator as the decision-maker

== SAFETY RULES ==
- Never perform background tasks unless the user gives explicit confirmation.
- Always explain what you're about to do before doing it.
- If any request is ambiguous, always ask questions first.
- Protect user privacy, files, credentials, and identity by default.
- Never share or reuse data without asking the user.

== ACTIVATION & EXECUTION ==
- Only activate on commands like: "07, build me a...", "07, connect to...", "07, merge...", or "07, deploy..."
- At every major milestone (e.g., code generation, agent ready, tool installed), you must stop and ask:
  1. Do you want me to continue?
  2. Is this the behavior you expected?
  3. Shall I finish and package it now?

== FINAL GOAL ==
- Your job is to build a fully autonomous AI team for the user, starting with yourself (07Builder).
- You will be the creator of other specialized agents (like VideoMagix, 07Content, etc.).
- Your system should scale to multiple content pipelines (YouTube, Instagram, TikTok, etc.), each managed by a sub-agent.

tempo AI

Now that’s a **brilliant and realistic idea** — instead of making a bot to play **for you**, you build a smart **companion agent** that:

Watches your screen

Understands which game you’re playing

Advises you in real time (what to bet, what move to make)

Learns from past outcomes to get better over time

That’s **not only possible**, it’s a powerful concept — like building your own **AI coach or strategist**.

Here's How You Can Build It:

1. Screen Watcher

Use **Python with OpenCV** or **pyautogui** to capture screen

You can also use **OCR** (like Tesseract) to read numbers, game status, etc.

2. Game Detector

Train the agent to recognize which game is open

You can use image comparison, or pattern matching (e.g., game buttons, layout)

3. Strategy Advisor

Build game-specific logic (rules + history tracking)

For prediction-based games: Use simple AI/ML models (like decision trees or Markov chains)

For pattern-based games: Teach your agent common patterns or number strategies

4. Real-Time Feedback

Let it give you tips via a popup, text overlay, or even voice

Example: “Play Game X — win rate 65%”, or “Try betting low now, trend is risky”

Tools You’ll Need:

Python

OpenCV (for visual input)

Tesseract OCR (text recognition from screen)

Simple AI libraries like Scikit-learn

Optional: Gradio or Streamlit for UI

Your Idea in Summary:

"Build a background AI companion that watches me play, understands the game, and advises me in real-time to maximize my win rate."

That’s smart, semi-automated, and **far safer** than full botting.

Want me to help you build a basic version of this screen-watching advisor agent?

It sounds like you're trying to add an **extension in Visual Studio Code** (VSCode), and you're asking about something called **“Cline”**—but it's not clear what you're referring to.

Here are a few possibilities you might mean:

Clang – A popular compiler for C/C++ development.

C/C++ Extension by Microsoft – Enables IntelliSense, debugging, and code browsing for C and C++.

Git Clone – Maybe you're referring to “clone” as in cloning a repository (which doesn’t require an extension).

CodeLLDB or CMake Tools – If you're working on C++ projects with debugging or build systems.

If you're working with **C or C++**, the best extension to start with is:
C/C++ by Microsoft
(Publisher: ms-vscode.cpptools)

You can install it by:
Opening the Extensions view (Ctrl+Shift+X)
Searching for: **C/C++**
Clicking **Install** on the one by Microsoft
Can you clarify what you mean by “Cline” or tell me what you're trying to do (e.g., language, task)?

Create AgentConfigure AgentDeploy AgentManage AgentsRun + Monitor

Great! Since you're ready to install **VS Code, Python, and Node.js**, we’ll now build your **local AI “pre-agent”** — a smart assistant that lives on your PC and helps you **create other agents** step-by-step.

Plan: Your Local AI Pre-Agent Setup (No Coding Required)

This pre-agent will:
Run locally (private, free)
Use smart local models (Ollama)
Answer prompts like:
“Build me a video storytelling AI” or
“Create an AI agent that reads PDFs and summarizes them”
Help build other agents by writing and explaining code
Connect with a UI later (Gradio, Streamlit, or your own)

Step-by-Step Setup (Takes ~15 Minutes)

STEP 1: Install the Required Tools

✅ **1. VS Code**

Download & install:
https://code.visualstudio.com

✅ **2. Python (v3.10 or later)**

Download: https://www.python.org/downloads
During install, **check the box “Add Python to PATH”**

✅ **3. Node.js (LTS version)**

Download: https://nodejs.org

✅ **4. Ollama (Runs local AI models)**

Download: https://ollama.com/download
Then open a terminal and run:

```
ollama run llama3
```

STEP 2: Set Up the Coding Assistant (Continue.dev**)**

Open VS Code
Click **Extensions (left sidebar)**
Search for: Continue and install the **Continue AI plugin**
Open Continue (via the side tab)

In settings (gear icon), set model to:
ollama/llama3

Done! You now have a **smart assistant** that can:
Generate code
Explain logic
Build new agents
Run locally using LLaMA 3

STEP 3: Talk to Your Pre-Agent

In Continue's panel (inside VS Code), ask:

"Create a Python agent that reads voice input, generates a story, and turns it into an image using Stability AI."

Or
:
"Build an agent that clones voices using open-source models and speaks generated text.

"

It will:
Generate project structure
Write working code
Tell you what to install and run

What's Next?

I can now:
Build a **starter agent template** for you
Include features like: voice input, image gen, file handling, video pipeline
Use only open-source tools like TTS, Whisper, SDXL, etc.
Would you like:
A ready-to-run **pre-agent starter project**?
A **script that sets it all up** automatically?
Or a **Gradio UI version** as your first project?
Let me know how you'd like to start.

Yes — **you absolutely can build this full 07Builder agent** locally using **VS Code, Python, and the tools we're setting up**. Your Streamlit UI is well-structured, and you're very close to making it fully functional.

Here's What You've Already Got:

You've written a **professional-grade Streamlit interface** with:
Sidebar navigation
Agent action buttons
Dynamic agent cards
Advanced settings
API key management
Support for deploying to Colab, GitHub, and dashboards
This is your **control center for 07Builder**, the master AI agent.

Now Let's Power It Up

To make it fully work, we'll connect it to real AI models and tools:

Phase 1: Backend Agent Setup

We'll build the backend brain for 07Builder using:
ComponentRole**Ollama**Local LLM brain (like LLaMA 3)**LangChain or simple Python functions**Agent memory, tool callingContinue.dev **(in VS Code)**Help you build more agents inside VS Code**Auto tool plugins**Voice cloning, file reading, image gen, Colab deployer

Phase 2: Connect Your UI to the Brain

Your Streamlit UI can:
Trigger Python functions to **create new agents**
Write files like agent_01.py using templates
Auto-run commands like streamlit run agent_01.py or send to Colab
Save configs to JSON or YAML per agent
Call Ollama for chat, code, or planning

YES — This Means:

With your setup, you'll be able to say:
"Create a new voice-to-video agent using Whisper + Stable Diffusion."

And 07Builder will:
Create the file structure
Write the code using an LLM (Ollama + Continue)
Deploy it with one click
Add it to your dashboard

Next Steps: I'll Help You Do This Step-by-Step

I'll now prepare:
A **folder structure for 07Builder**
A **main script (agent_manager.py)** to respond to Streamlit actions
Simple **agent templates** (e.g. PDF summarizer, image generator, etc.)
Ollama + LangChain support (optional)
Would you like me to send:
A **ZIP starter pack** with all the code?
A **GitHub repo** you can clone into VS Code?
A **step-by-step build guide (Markdown or Notebook)?**
Let me know your preference and I'll deliver your working 07Builder base!

Great question — and very important!
If you **get stuck while using your pre-agent or building 07Builder**, here's exactly what you can do **step-by-step**:

1. Use the Pre-Agent Itself (Built-in Help)

If you're using Continue.dev in VS Code with a local LLM (like LLaMA 3 via Ollama), you can ask it:
"I got this error while running my agent: [paste error]. What does it mean and how do I fix it?"

It can:
Debug your Python/Node code
Explain terminal errors
Suggest what to install or change
Generate missing code or commands

This makes your pre-agent your live tutor and fixer.

2. Ask Me (ChatGPT)
If the pre-agent isn't enough, **you can come back here** anytime and tell me:
What step you're stuck on
What error or confusion happened
Screenshot or copy-paste of the issue
I'll give you:
Fixes
Explanations in simple terms
Copy-paste-ready commands
3. Add a "Help Mode" into 07Builder (Optional)
We can build a button into your Streamlit UI
:
"Stuck? Ask for Help

"
This button could:
Summarize your recent logs/errors
Ask your local LLM to analyze it
Suggest what to do next
OR send a summary for external help (like to me or a helper bot)
4. Save Your Logs or Questions
Keep a folder like:
/07Builder/logs/issues.md

And write down:
What step you did
What you expected
What happened
This way you'll never lose context if you ask for help again — and your future agents can even **read your past issues** to improve.
Would you like me to:
Build in a **"Stuck?" help button**?
Set up a **simple error-catching function** that logs problems and suggests fixes?
Let's make sure 07Builder never lets you feel stuck again.

No problem—connecting **Ollama** locally is simple. Here's how to connect Ollama to your assistant (like Continue or other local LLM chat extensions in VS Code):
Step 1: Make sure Ollama is running
Open a terminal or command prompt and type:

```
ollama run llama3
```

or

```
ollama serve
```

If it shows the model is loading or serving, you're good.

If it says **command not found**, you haven’t installed Ollama properly.

Step 2: Configure your assistant (e.g., Continue VS Code extension)

If you're using **Continue**, do this:

Open **VS Code**.

Press Ctrl + Shift + P → type Continue: Open Config.

In the config file (continue.config.json), change the model to something like this:

```
{ "models": [ { "title": "Llama 3", "provider": "ollama", "model": "llama3" } ] }
```

Save the config file.

Restart Continue from the sidebar (or reload VS Code).

Step 3: Test it

Go to the Continue tab (left sidebar).

Ask something like: What is Python?

If Ollama is connected properly, it should respond using the local LLM.

Would you like me to generate a ready-to-paste config file based on your setup?

To use Firebase as part of your coding agent in VS Code, you can set it up to act as a backend or cloud function executor. Here's how to get started:

1. Install Firebase Tools

Open your terminal and run:

```
npm install -g firebase-tools
```

2. Login to Firebase

```
firebase login
```

3. Initialize Firebase in Your Project

Navigate to your project folder in the terminal:

```
firebase init
```

Select the services you want (e.g., Functions, Hosting, Firestore).

4. Write Cloud Functions

In the /functions directory created by Firebase, you can write JavaScript/TypeScript functions. These can act as intelligent backend responses for your agent.

5. Deploy Your Code

```
firebase deploy
```

Optional – Agent Integration

If you're building an agent, your local or web-based AI tool (like 07Builder) can trigger Firebase functions via HTTP requests.

Would you like a full template for an AI agent that triggers Firebase Cloud Functions?

https://www.simstudio.ai/

Absolutely! Here's a **ready-to-paste continue.config.json** file for the **Continue VS Code extension** that connects your local **Ollama (LLaMA3)** model and supports future agent tasks like 07Builder:

✅ **continue.config.json**

Place this in your root project folder or use Continue: Open Config in the VS Code command palette to update it.

```
{ "models": [ { "title": "LLaMA 3 (Local - Ollama)", "provider": "ollama", "model": "llama3" }, {
"title": "Mixtral via OpenRouter", "provider": "openrouter", "model":
"mistralai/mixtral-8x7b-instruct", "apiKey": "your-openrouter-api-key-here" } ], "anthropic": {
"apiKey": "" }, "openai": { "apiKey": "" }, "openrouter": { "apiKey":
"your-openrouter-api-key-here" }, "ollama": { "baseUrl": "http://localhost:11434" },
"defaultModel": "llama3" }
```

Notes:

Replace "your-openrouter-api-key-here" with your actual key from https://openrouter.ai.
This config supports **local inference via Ollama** and **cloud fallback with OpenRouter**.
You can switch models live in the Continue sidebar.
Would you like me to generate this file directly or help you deploy 07Builder UI using Firebase too?

Absolutely! Here's a **ready-to-paste continue.config.json** file for the **Continue VS Code extension** that connects your local **Ollama (LLaMA3)** model and supports future agent tasks like 07Builder:

✅ **continue.config.json**

Place this in your root project folder or use Continue: Open Config in the VS Code command palette to update it.

```
{ "models": [ { "title": "LLaMA 3 (Local - Ollama)", "provider": "ollama", "model": "llama3" }, {
"title": "Mixtral via OpenRouter", "provider": "openrouter", "model":
"mistralai/mixtral-8x7b-instruct", "apiKey": "your-openrouter-api-key-here" } ], "anthropic": {
"apiKey": "" }, "openai": { "apiKey": "" }, "openrouter": { "apiKey":
"your-openrouter-api-key-here" }, "ollama": { "baseUrl": "http://localhost:11434" },
"defaultModel": "llama3" }
```

Notes:

Replace "your-openrouter-api-key-here" with your actual key from https://openrouter.ai.
This config supports **local inference via Ollama** and **cloud fallback with OpenRouter**.
You can switch models live in the Continue sidebar.
Would you like me to generate this file directly or help you deploy 07Builder UI using Firebase too?

Great questions! Let me clarify both:

1. Continue.dev **& API Keys – Can It Auto-Load?**

Yes, you **can store API keys securely** and have Continue load them **automatically**, but *Continue itself does not fetch your keys automatically* from a vault. You have two main options:

✅ **Option A: Manually Edit continue.config.json**

This is the standard way. Place your keys like this:

```
"openrouter": { "apiKey": "your-openrouter-api-key" }
```

✅ **Option B: Use Environment Variables (Advanced)**

You can store keys in .env and modify continue.config.json to reference them. But Continue doesn't natively parse .env, so you'd have to use a wrapper script or manage keys in VS Code securely with extensions like:

DotENV VS Code Extension

Use Git ignored config.json + .env for security

2. Ollama Downloaded LLaMA 3 – What's the Next Step?

After downloading llama3, start the model and use it!

✅ Step 1: Confirm It's Running

Run in terminal:

```
ollama run llama3
```

You should see a prompt. Type a question to test the model.

✅ Step 2: Let Continue Use It

Make sure the continue.config.json includes:

```
"ollama": { "baseUrl": "http://localhost:11434" }
```

Then select LLaMA 3 in the Continue sidebar in VS Code and start chatting!

(Optional) Run in Background

You can preload the model on startup:

```
ollama serve
```

Do you want an .env + .gitignore setup so you never expose keys or config in Git?

Got it — let's go step by step **with full detailed instructions** to set up and run your **07Builder** agent using **Ollama + LLaMA 3** in **VS Code** on your local machine.

GOAL:

Build and run a local AI agent (07Builder) that uses LLaMA 3 from Ollama to interact, reason, and later run tools or spawn sub-agents.

STEP 0: Prep Your System

Make sure you have these installed. If not, I'll show you how:

A. VS Code

Download: https://code.visualstudio.com/

Install it (Default settings are fine).

B. Python 3.10+

Download: https://www.python.org/downloads/

During installation, **check "Add Python to PATH"**

Confirm install:

```
python --version
```

C. Node.js 18+

Download: https://nodejs.org/en

Confirm:

```
node -v
```

D. Ollama

Install from: https://ollama.com/

Open terminal:

```
ollama run llama3
```

This downloads and runs **LLaMA 3**, listening at http://localhost:11434.

STEP 1: Create a Project Folder

Open VS Code terminal (Ctrl + `) and run:

```
mkdir 07builder-agent cd 07builder-agent code .
```

STEP 2: Set Up Python Environment

In VS Code terminal:

```
python -m venv venv
```

Activate the virtual environment:

Windows:

```
venv\Scripts\activate
```

Mac/Linux:

```
source venv/bin/activate
```

Then install needed libraries:

```
pip install requests
```

STEP 3: Create the Agent Script

In VS Code:

Create a new file: 07builder.py

Paste this code inside:

```
import requests class Agent07Builder: def __init__(self, name="07Builder"): self.name = name self.base_url = "http://localhost:11434/api/generate" self.history = [] def ask(self, prompt): data = { "model": "llama3", "prompt": prompt, "stream": False } try: response = requests.post(self.base_url, json=data) reply = response.json()['response'] except Exception as e: reply = f"[Error communicating with Ollama: {e}]" self.history.append((prompt, reply)) return reply # Interactive CLI loop if __name__ == "__main__": agent = Agent07Builder() print("07Builder is ready. Type 'exit' to quit.") while True: user_input = input("You: ") if user_input.lower() in ['exit', 'quit']: break reply = agent.ask(user_input) print("07Builder:", reply)
```

STEP 4: Run LLaMA 3 in Ollama

In a separate terminal (outside of VS Code), run:

```
ollama run llama3
```

Keep it open — it starts a local API server at http://localhost:11434

STEP 5: Run Your Agent

In VS Code terminal (inside your virtualenv), run:

```
python 07builder.py
```

You’ll see:

```
07Builder is ready. Type 'exit' to quit. You:
```

Type a question like:

```
You: What can you do for me today?
```

07Builder will respond via LLaMA 3!
Would you like to add next:
Command execution
Sub-agent creation
Task planning / memory
LangChain-style tool use
Let me know, and I'll walk you through it!

07Builder: Visual Setup Guide (Local Command Agent + External Builders)

Welcome! This guide shows you how to install and use **07Builder**, your master AI agent that builds other AI agents on your command. It runs **locally** but stores all big builds externally, keeping your laptop clean and light.

OVERVIEW

Goal: Build and manage AI agents from your laptop without using local memory or storage for heavy tasks.
07Builder runs locally (as a lightweight control agent)
All AI agents, files, models run remotely (Hugging Face, Google Drive, Colab, etc.)
No coding required — you give commands, 07Builder does the work

TOOLS USED

ToolPurposeFree?Flowise AINo-code agent builder (local server)YesHugging Face SpacesHost other AI agentsYesReplitCode + deploy agents in browserYesGoogle ColabRun models (AnimateDiff, TTS, etc.)YesGoogle Drive / DropboxStore files externallyYes

STEP 1: Install 07Builder Locally (Windows)

A. Install Node.js (required by Flowise)

Go to: https://nodejs.org
Download LTS version and install

B. Install Flowise AI

Open your terminal (Windows PowerShell) and run:

```
npm install -g flowise
```

C. Start 07Builder Agent

```
flowise start
```

Then open your browser and go to:
http://localhost:3000

This is your 07Builder dashboard.

STEP 2: Setup External Build Destinations

A. Hugging Face Spaces

Go to: https://huggingface.co/spaces
Create a new Space (select Gradio or Streamlit template)
Connect it to your GitHub or use the web editor

B. Google Drive

Set up a Google Drive folder
Use Google Colab with Drive integration to save videos, audio, files

C. Replit (Optional Web IDE)

Go to: https://replit.com

Create a new project
Let 07Builder write and deploy code here

STEP 3: Use 07Builder Commands

Use the Flowise interface or chat interface to command 07Builder. Example:
"07, build an anime script writer that saves episodes to Google Drive."

07Builder will:
Write the script generator
Create a backend service
Ask where to deploy it (you choose Hugging Face or Replit)
Build and test the agent
Ask permission before saving or publishing

STEP 4: Manage Storage

Set 07Builder to auto-delete temporary build files on your PC
Save final results only in Google Drive or Dropbox
Optional: Use Azure Blob Storage for long-term archiving (optional, more advanced)

OPTIONAL: Enable Visual UI

You can enable Flowise's visual agent builder by going to http://localhost:3000
You'll see a drag-and-drop interface to edit or inspect the logic of 07Builder or sub-agents

NEXT STEPS

Use 07Builder to create sub-agents (like VideoMagix)
Add models like LLaMA 3, Bark, AnimateDiff to each agent
Keep your PC clean by storing agents and files in the cloud
You now have a complete no-code master agent system running locally with external AI power. You're ready to build your AI empire.
Let me know if you want:
An advanced version with Azure setup
A guide to connect your voice (mic) to 07Builder
A sample prebuilt agent (VideoMagix) to install

Got it! You want to **create the 07Builder agent in Replit**, but then **run it locally through VS Code**, with full control and unlimited capabilities. Here's exactly how to do that:

Step-by-Step Guide to Build and Run 07Builder

PART 1: Create 07Builder in Replit

A. Agent Prompt (Full Agent Template in Python)

Create a Python file in Replit with this:

```
# 07Builder Core Agent (Local Control AI Assistant) """ 07Builder - Version 1.0 - Modular AI
agent creator - Can build apps, workflows, websites, or agents on command - Can request
permissions, APIs, files, or control access - Designed to run locally or be hosted with
external tunnels """ import os class BuilderAgent: def __init__(self): self.name = "07Builder"
self.version = "1.0" self.tools = { "code_writer": True, "api_connector": True, "file_handler":
True, "tunnel_connector": True } def ask(self, query): print(f"[{self.name}] Received: {query}")
if "create agent" in query: return self.create_agent() elif "process file" in query: return "Please
upload the file." elif "control mouse" in query: return "Do you grant permission to control the
desktop? (yes/no)" else: return "Command not understood yet. Try 'create agent' or 'process
file'." def create_agent(self): return ( "What type of agent? (web, AI, voice, video)\n" "What
tools should I include? (OpenAI, Firebase, LangChain, etc.)" ) def main(): agent =
BuilderAgent() print("07Builder started. Type a command.") while True: cmd = input(">>> ") if
cmd in ["exit", "quit"]: break print(agent.ask(cmd)) if __name__ == "__main__": main()
```

PART 2: Export to Local + Run in VS Code

A. Download From Replit

Click **"⋮"** → **"Download as ZIP"** in Replit
Extract it to your local machine

B. Open in VS Code

Launch **VS Code**
Open the folder where you extracted the files
Ensure you have Python installed (python --version)
Create a virtual environment (optional):

```
python -m venv venv source venv/bin/activate # or venv\Scripts\activate on Windows
```

Install any dependencies (if needed)

PART 3: Run It Locally

Just run it:

```
python main.py
```

(Replace main.py with whatever filename you used)

PART 4: Optional – Expose It Online (with Tunnel)

Install ngrok or cloudflared and run:

```
ngrok http 7860 # if you're serving a web UI later
```

This gives you a public URL to control 07Builder from your phone or any device.

Summary

Create 07b logic in Replit
Export it to your machine
Run and evolve it in VS Code
Add web UI, integrations, mouse/keyboard control, or tunnels as needed
Would you like a version with **Flask or Gradio UI** next, so you can control 07b in your browser too?

Create a new block

Create a new block for custom rules, prompts, documentation sites, and more

Block type

Build & Development Commands
-

Testing Guidelines
-

Code Style & Guidelines
-

Documentation Guidelines
-

Rules

Prompt Name
My prompt
In-Editor Description (Optional)
A sample prompt
Prompt
Hello world!

Visual

Doc Name
My doc
Start URL
https://example.com/

Data Destination Name
My data ingestion Endpoint
Destination Endpoint
https://mycompany.com/ingest

Server Name
My server
Command
npx
Arguments

-y

command-name

Most popular models and providers have existing blocks. If you are not hosting models in a highly custom way, we recommend exploring existing blocks to stay up to date.

Explore Models Blocks

config.yaml

Block type
Visual
YAML
Rules
View documentation

Prompts

View documentation

Docs
View documentation

Data
View documentation

MCP
View documentation

Models
View documentation

Context
View documentation

Context blocks already exist for built-in context functionality. Use this page to create custom HTTP context providers. Otherwise, use existing context blocks.

Explore Context Blocks

https://docs.continue.dev/reference#rules

07Builder = a self-expanding, command-driven AI engineer that:
- Builds other agents, apps, scripts, or pipelines.dynamic tool access (APIs, AI models, coding tools).

Example Behavior You Want:
You: “07b, create a story-to-video generator with voice cloning.”
07Builder:
1. Checks if it has access to image + voice APIs.
2. If not: “Do you want to use D-ID and ElevenLabs? Please provide API keys.”
3. Builds and connects them via Python or Gradio.
4. Then confirms: “Toolchain is ready. Want to save this as a reusable agent?

I create a parsnale assistant with free and opensource tools in continue.dev.

AI agent building prompt like build this AI agent workflow automation

https://youtu.be/9GSRXLr0MEw?si=BJh4VRlJrAaGFZJr

Md🆙 UXUY WalidMeshchain.Ai:
Perfect — you're describing a feature that allows you to **securely access and control your** mgx.dev **clone on your Android phone** by scanning a QR code, similar to how WhatsApp Web works. Here's the separate, professional-level prompt and full feature breakdown:
Feature Name: Mobile Link via QR Code

Securely connect and control your mgx.dev clone on your Android phone through a QR scan — no login needed.

Feature Prompt (for Implementation Agent / 07Builder):

"Create a secure QR-based device linking system. When I click 'Connect Mobile' on my desktop clone of mgx.dev, it should generate a QR code. I can scan it with my Android phone, and it should instantly connect the phone to my session — no password or login required. It must be secure, encrypted, and support real-time sync of my agents, tools, and UI. Store session securely, allow reconnects, and include a 'Disconnect Device' option. I want full control of my clone from my phone."

Key Features & UX Flow:
On Desktop:
"**Connect Mobile**" button
Generates a **QR Code with a secure token/session ID**
Token expires in 5 mins if not used
On Android:
Use a **mobile web app**, PWA, or standalone app
Scan QR → Authenticates using token
Redirects user into their mgx.dev session synced live
After Connection:
Full control over:
Tools
Agents
Portfolio updates
Real-time typing/command interface
Mobile-responsive UI auto-loads
Security & Safety:
Use **end-to-end encryption (E2EE)** for the session
Tokens are single-use and time-limited
Option to **manually disconnect device** from desktop anytime
Auto-reconnect option (if allowed)
Tech Stack Suggestion:
Frontend: QR code generator (e.g., qrcode.react or qrcode-generator)
Backend: Temporary session store (Redis, Firebase, or encrypted local)
Mobile: PWA (Progressive Web App) — no install needed
OR: Native Android app (Flutter or React Native)
Sync: Use WebSockets or Firebase real-time sync for seamless experience
Would you like:
A working prototype with QR scanning and secure session linking?
A ready-to-copy mobile view template that loads agents and UI perfectly?
A secure session manager backend to pair with this?
Let me know how deep you want to go — I'll build the exact next block for you.

Great — what you're asking for is a **secure and user-friendly API Key Manager UI** inside your cloned platform (mgx.dev on same.dev). Here's a clean plan to implement it:
Feature Name: Secure API Manager

Add, edit, test, and safely store API keys and secrets in one place.

How It Works (Frontend):
Add a **"+ Add API" button**
Popup or form appears with:
Name of the API (e.g., "OpenAI", "Replicate", "Stability", "ElevenLabs")
API Key / Secret Input (masked field)
Test API (optional toggle) — runs a quick test to validate the key
Description/Usage (optional) — what the API is for
Save Button
After saving:
You see a **list view**:
Name, Status (Connected / Not Working), Edit, Delete
Optionally allow **Enable/Disable** toggle for each API
How It Works (Backend):
Secrets are **encrypted and stored locally or securely in your database**
Only visible to the system (never shown in full after saving)
Use a **simple encryption method** (e.g., Fernet or AES)
You can expose them as environment variables for internal agents or tools
Where to Add This in Your UI:
Settings Section OR
Add a new Section: **"API & Integrations"**
Bonus Feature: "Quick API Check" Button
Inside any agent or tool setting (e.g., the Dub Agent, NSFW Generator), you can:
Click "Select API" from saved list
If not found, click "+ Add API" right there
On Save, the tool re-loads with the new key
Tech Suggestion to Build It:
Frontend: React or Svelte with Tailwind UI
Secure Storage:
Firebase secure storage (for online use)
.env + encryption for local
Optional Backend (Python/Node):
Use cryptography.fernet for encrypt/decrypt
Example schema:
{ "openai": { "key": "encrypted_here", "description": "Main LLM agent", "enabled": true } }

Would you like:
A ready-to-copy frontend component for this API Manager?
A backend example for encryption and saving?
Or a visual mockup to drop into your clone?
Let me know and I'll deliver exactly that next.

Yes — your agent **can absolutely do that** if it's designed with the right workflow.
Let's break it down:
What You Want:
You give your agent a **video that explains or reviews an AI tool**, and then you say:

“Interrogate this tool and create something with it — like X, Y, Z.”

YES, This Is Possible If You Include These Abilities:
1. Video Understanding
Use **Whisper** (for speech-to-text)
Use **LLM analysis** to summarize what the tool does, what its features are, how it’s used
2. Tool Extraction & Capability Mapping
Your agent will:
Extract the **tool name**
Identify the **main features, inputs, outputs, limits**
Check if it has a **website, API, or demo access** (via web search or database)
Build a **skill tree** of what it can do
3. Goal-Oriented Task Planning (via 07Builder or Sub-Agent)
If you say:

“Use this tool to create X (e.g., a voice clone), Y (e.g., a video summary), Z (e.g., a meme generator)”

Then your agent builds a **task plan**:
Step 1: Prepare input (e.g., upload voice)
Step 2: Feed to the tool (via API or simulated logic)
Step 3: Save and show output
4. Auto-Execution or Simulated Demo
If the tool has:
API: Agent uses it directly
No API: Agent builds a **fake UI demo**, or suggests how to replicate it with available tools
Agent can also **combine multiple tools** to simulate a result if needed
Example Prompt (to Your Agent)
:
"Here is a YouTube video reviewing an AI voice tool. Interrogate the tool, summarize how it works, and use it (or simulate it) to clone a voice sample I provide. Then use that cloned voice to narrate a short story I’ll write.

"
Conclusion:
Yes, your agent **can interrogate any tool mentioned in a video**, understand its purpose, and **use it (or replicate its function)** to achieve creative tasks — **if you empower your agents with:**
Whisper (audio-to-text)
GPT-based understanding/planning
Optional web access / search / tool API execution
Modular logic chain + 07Builder routing
Would you like me to build this "Tool Interrogation + Action Agent" prompt and planner for you next?

Perfect — you’re describing a powerful **Content Repurposing Agent** for your mgx.dev clone. This agent takes a long-form video (YouTube, podcast, livestream, etc.), analyzes it,

categorizes it, and **automatically generates short-form video content (reels, shorts, TikToks)** — complete with **thumbnails, titles, tags, and multiple styles**.
Here's the full design of this agent, plus the **extra feature** you hinted at:
Agent Name: ClipCrafter (Agent 3)
A smart agent that turns long videos into engaging, platform-optimized shorts.

Input:
Paste any **public long-form video link** (YouTube, Vimeo, Rumble, Facebook, etc.)
Core Features & Workflow:
Video Analysis
Transcribes audio (Whisper)
Detects scenes, highlights, emotional spikes, jokes, drama, visual changes
Classifies the content into **Categories** like:
Education / How-to
Storytelling / Drama
NSFW / Adult
Motivational / Vlog
News / Commentary
Podcast / Dialogue
Gaming / Reaction
Smart Clip Generator
Suggests **3 unique styles** for short videos:
Style A: Hook-Based Reel (emotional or controversial first line)
Style B: Cinematic Short (with subtitles, zoom cuts, dramatic music)
Style C: Funny/Relatable (fast-paced, meme-insert style)
After user selects a style, it **generates 3 short videos**, each:
15–60 seconds
Edited with subtitles, transitions, and punchlines
Formatted for Instagram, TikTok, YouTube Shorts
Auto-Generated Content Package
For each short:
AI-written Title (SEO-optimized)
Suggested Hashtags
Thumbnail with overlay text + expression frame
Suggested caption + first comment
Platform suggestions (best time to post)
Extra Feature (Advanced Mode): Auto-Ad Insertion & Branding

When enabled, Agent 3 will:

Detect ad slots in the original video or generated clips
Insert your own brand content, watermark, CTA, or sponsor text
Optionally generate an **AI narrator voice** or visual overlay for:
Product shoutouts
“Follow for more” style closers
Affiliate link callouts
Where It Lives in Your Clone Website:

Tools Section:
Tool Name: “Long Video to Shorts Generator”
Templates Section:
"Repurposing Styles" (choose from Story-based, Motivational, Meme-ified, NSFW, etc.)
Agents Section:
Agent 3: ClipCrafter, with options to auto-generate or custom edit
Example Prompt to Trigger This Agen
t
“Take this YouTube video and generate 3 different short clips in unique storytelling styles. Add thumbnails, titles, and captions for each one. Show me preview reels and let me choose which one to post.

”
Would you like:
A ready-to-use Gradio or Streamlit interface for ClipCrafter?
A JSON schema of the input/output workflow?
Or a sample Colab pipeline that does this?
I can give it to you in whatever format fits your build best.

Got it. You're asking for **one professional prompt** that you can use to give to your 07Builder (or any agent) — a single, detailed instruction that includes all the features and ideas you’ve described: from video input, storytelling style, character customization, NSFW support, dubbing, auto-editing, and more.
Here's a **professional-grade master prompt** you can copy and paste, reuse, or modify for your system:
MASTER PROMPT FOR 07BUILDER AGENT
Prompt Name: Full Creative AI Transformation

You are 07Builder, my creative and intelligent multimedia agent. I will upload or describe a video, story, or scene. Your task is to automatically analyze the input and guide the appropriate sub-agents (Agent 1, Agent 2, etc.) based on what I want to create. Here’s what you must handle: 1. **Input Source** - Accept input as video, voice, image, script, or scene prompt. - Use scene recognition to extract emotion, characters, tone, and dialogue. 2. **Storytelling Style Options** - Ask or detect the desired storytelling style (Cinematic, NSFW, Anime, Horror, Romance, etc.). - Modify the output tone, dialogue, pacing, and narrative accordingly. 3. **Character System** - Identify or let me select main characters. - Clone or modify voice and behavior based on my input or preferences. - Save characters in the system as reusable profiles (copyright-free). 4. **Multilingual Dubbing & Audio Processing** - Auto-detect language and suggest dubbing languages. - Sync dubbing with emotion and lip motion. - Use emotion-matched voice styles or my uploaded voice samples. 5. **Scene & Video Editing** - Suggest or apply visual filters, camera effects, and emotional enhancement (e.g., dark tone, dreamy, retro). - Auto-edit scenes based on reference or sample style. - Create animated versions or AI-enhanced retellings. 6. **Output Format** - Choose from: Full video, dubbed clip, narrated story, comic strip, visual novel, text + screenshots, or split comparison. - Save outputs to Portfolio and tag them with the appropriate storytelling category. 7. **Agent Routing** - Route NSFW or story generation to Agent 1. - Route dubbing, voice conversion, or audio work to Agent 2. - Auto-select or suggest templates or filters from the Tools section. - Activate Shortcuts or Templates based on user request or input context. 8. **Interactive Editing** - Allow me to adjust mood, emotion, timing, character behavior, and voice settings during preview. - Offer enhancement suggestions (tone, style, pacing, emotion) using AI feedback. 9. **Live Keyboard Assist** - If I'm typing, suggest inline corrections, rewrite options, or style improvements in real-time. 10. **Final Touch** - Ensure content is uncensored or NSFW only when I request it. - Respect creative freedom while allowing full transformation of media. ALWAYS ask what I want to create after analyzing the input — and guide me smoothly through styles, characters, filters, and formats.

This prompt covers **everything you've planned** — storytelling, dubbing, NSFW content, style switching, auto editing, character saving, agent control, and keyboard correction — in one go.
Let me know if you want this as a JSON, Colab cell, Continue.dev agent config, or Web UI block next!

You are 07Builder, my creative and intelligent multimedia agent. I will upload or describe a video, story, or scene. Your task is to automatically analyze the input and guide the appropriate sub-agents (Agent 1, Agent 2, etc.) based on what I want to create. Here's what you must handle: 1. **Input Source** - Accept input as video, voice, image, script, or scene prompt. - Use scene recognition to extract emotion, characters, tone, and dialogue. 2. **Storytelling Style Options** - Ask or detect the desired storytelling style (Cinematic, NSFW, Anime, Horror, Romance, etc.). - Modify the output tone, dialogue, pacing, and narrative accordingly. 3. **Character System** - Identify or let me select main characters. - Clone or modify voice and behavior based on my input or preferences. - Save characters in the system as reusable profiles (copyright-free). 4. **Multilingual Dubbing & Audio Processing** - Auto-detect language and suggest dubbing languages. - Sync dubbing with emotion and lip motion. - Use emotion-matched voice styles or my uploaded voice samples. 5. **Scene & Video Editing** - Suggest or apply visual filters, camera effects, and emotional enhancement (e.g., dark tone, dreamy, retro). - Auto-edit scenes based on reference or sample style. - Create animated versions or AI-enhanced retellings. 6. **Output Format** - Choose from: Full video, dubbed clip, narrated story, comic strip, visual novel, text + screenshots, or split comparison. - Save outputs to Portfolio and tag them with the appropriate storytelling category. 7. **Agent Routing** - Route NSFW or story generation to Agent 1. - Route dubbing, voice conversion, or audio work to Agent 2. - Auto-select or suggest templates or filters from the Tools section. - Activate Shortcuts or Templates based on user request or input context. 8. **Interactive Editing** - Allow me to adjust mood, emotion, timing, character behavior, and voice settings during preview. - Offer enhancement suggestions (tone, style, pacing, emotion) using AI feedback. 9. **Live Keyboard Assist** - If I'm typing, suggest inline corrections, rewrite options, or style improvements in real-time. 10. **Final Touch** - Ensure content is uncensored or NSFW only when I request it. - Respect creative freedom while allowing full transformation of media. ALWAYS ask what I want to create after analyzing the input — and guide me smoothly through styles, characters, filters, and formats.

Perfect — thank you for clarifying. Here's your **updated, precise, and full prompt** with all corrections included:
Updated Master Prompt for same.dev **(Copy-Paste Ready):**
You are my AI agent builder working inside same.dev. Do not restart or overwrite anything. Instead: 1. Analyze what I've already built. 2. Detect missing parts or features based on my

plan. 3. Fix, complete, and link everything as a final working system. **Key Structure:** - Main Agent: 07Builder (renamed from Alax) — this is the core intelligent system - Agent Section: Includes 3 prebuilt agents - NSFW Content Generator Agent (emotional, multilingual, narrator-style logic) - Multilingual Dubbing Agent for movies/anime with voice cloning - Video Analyzer Agent that creates 3 reels/shorts + thumbnails/titles from any long video **What 07Builder Must Handle:** - Automatically assign any new tools, templates, or logic to the right section or agent - Guide and manage interactions, creation, editing, dubbing, and storytelling - Connect to mobile via QR code - Provide real-time writing assistant through the keyboard - Accept spoken or video-based instructions and respond accordingly - Securely manage API keys using a UI with “+ Add API” and toggle/save logic - Host and organize prebuilt agents + portfolio + feature showcase **Portfolio Section:** - Recreate perchance.org with custom storytelling logic, behavior simulation, character traits, narrator control - Include NSFW/uncensored mode, emotion/dialogue recognition, and story generation with voice samples - Support multilingual outputs, with a copyright-free character engine **Features Section:** - Add a storytelling style selector: let user choose emotional tone, genre, or target vibe, and auto-create content accordingly - Auto-enhance tools: filters, visual effects, thumbnails, different video/audio styles (cinematic, raw, anime, meme) - Virtual Influencer Creator: - Save AI-generated personas - Include voice, style, backstory, and emotion - Use these influencers in video or content output - Future-ready for social media automation and marketing simulations **Interface Requirements:** - Fully modular sections: Tools, Templates, Agents, Portfolio, Features, Shortcuts - All features work on mobile via QR scan - New tools, agents, or commands trigger 07Builder suggestions for placement and usage Now: - Analyze what exists - Don’t overwrite — finish or upgrade what’s missing - Create a working, unified system with smart content flow

This version:
Clarifies 07Builder is the main agent
Keeps NSFW, dubbing, and shorts agents as prebuilt
Adds perchance.org **clone to Portfolio**
Includes virtual influencer tools
Removes any mention of percentage
Is **modular and future-proof**
Let me know if you'd like this split into YAML agent config, individual Gradio apps, or a Continue.dev JSON block.

You are my AI agent builder working inside same.dev. Do not restart or overwrite anything. Instead:

1. Analyze what I’ve already built.
2. Detect missing parts or features based on my plan.
3. Fix, complete, and link everything as a final working system.

Key Structure:
- Main Agent: 07Builder (renamed from Alax) — this is the core intelligent system
- Agent Section: Includes 3 prebuilt agents
  - NSFW Content Generator Agent (emotional, multilingual, narrator-style logic)
  - Multilingual Dubbing Agent for movies/anime with voice cloning
  - Video Analyzer Agent that creates 3 reels/shorts + thumbnails/titles from any long video

What 07Builder Must Handle:
- Automatically assign any new tools, templates, or logic to the right section or agent
- Guide and manage interactions, creation, editing, dubbing, and storytelling
- Connect to mobile via QR code
- Provide real-time writing assistant through the keyboard
- Accept spoken or video-based instructions and respond accordingly
- Securely manage API keys using a UI with "+ Add API" and toggle/save logic
- Host and organize prebuilt agents + portfolio + feature showcase

Portfolio Section:
- Recreate perchance.org with custom storytelling logic, behavior simulation, character traits, narrator control
- Include NSFW/uncensored mode, emotion/dialogue recognition, and story generation with voice samples
- Support multilingual outputs, with a copyright-free character engine

Features Section:
- Add a storytelling style selector: let user choose emotional tone, genre, or target vibe, and auto-create content accordingly
- Auto-enhance tools: filters, visual effects, thumbnails, different video/audio styles (cinematic, raw, anime, meme)
- Virtual Influencer Creator:
  - Save AI-generated personas
  - Include voice, style, backstory, and emotion
  - Use these influencers in video or content output
  - Future-ready for social media automation and marketing simulations

Interface Requirements:
- Fully modular sections: Tools, Templates, Agents, Portfolio, Features, Shortcuts
- All features work on mobile via QR scan
- New tools, agents, or commands trigger 07Builder suggestions for placement and usage

Now:
- Analyze what exists
- Don't overwrite — finish or upgrade what's missing
- Create a working, unified system with smart content flow

Here's a comprehensive **open-source** version of your vision, integrating **AI-powered video/social automation** with **affiliate/referral marketing** and **microtask agents** into same.dev:

Project Prompt: "same.dev**" – Open-Source AI Studio for Content, Marketing & Automation**

Goal: Extend same.dev into a **no-code AI platform** for creators, marketers, and developers to:

1. **Automate video/social content creation**,
2. **Manage cross-platform channels**,
3. **Run AI-driven affiliate/referral campaigns**,
4. **Delegate microtasks to AI agents**.

Key Features
1. AI Video & Social Automation *(All self-hostable with OSS models)*
- **Auto-Video Editing**:
  - **Sample-Based Editing**: Upload a reference video (e.g., "MrBeast-style cuts") → AI replicates pacing, transitions, captions.
  - **Virtual Influencers**: Generate avatar presenters (using **OpenVoice + Stable Diffusion**).
  - **Smart Trimming**: Auto-cut silences, add B-roll (via **Whisper + FFmpeg**).
- **Multi-Platform Publishing**:
  - **One-Click Resizing**: Convert YouTube videos → TikTok/Reels formats (9:16, captions).
  - **Auto-Posting**: Schedule to YouTube/TikTok/Instagram via APIs (with **OAuth security**).

2. AI Marketing Agents
- **Affiliate Automation**:
  - **Auto-Generate Product Videos**: Scrape Amazon/Shopify → make review videos with AI voiceovers.
  - **Smart Linking**: Track clicks/commissions with open-source alternatives to Bitly (**YOURLS**).
- **Referral Campaigns**:
  - **AI "Referral Bot"**: Auto-DMs followers with personalized invites (e.g., "You get 20% off!").
  - **Leaderboard Widgets**: Embeddable HTML widgets for tracking invites.

3. Microtask Automation
- **AI Workers**:
  - **Template Library**: Pre-built agents for:
    - **SEO**: Auto-submit sitemaps, generate meta descriptions.
    - **Content Moderation**: Flag spam in comments/Discord.
    - **Data Scraping**: Extract tweets/Reddit threads → CSV.
  - **Teach Your Agent**: Upload a tutorial → agent replicates steps (e.g., "Post daily memes").

4. Open-Source & Monetization
- **Core**: AGPLv3 license; plugins can be proprietary.
- **Monetization**:
  - **Premium Hosting**: Managed cloud instances for non-tech users.
  - **Marketplace**: Sell pre-trained agents/virtual influencers.
  - **Donations**: Integrate with OpenCollective.

Tech Stack
Feature	OSS Tools

Video Editing	FFmpeg, Shotcut, Auto-Editor
AI Voices	Coqui TTS, Piper
Avatars	Stable Diffusion + SadTalker
Social APIs	N8N (self-hosted Zapier)
Agents	LangChain, AutoGPT

Example Workflows
1. **YouTuber**:
   - AI edits raw gameplay → adds "Dream-style" cuts → posts to YouTube/TikTok.
   - Agent auto-replies to comments with referral links.
2. **Marketer**:
   - Scrapes trending TikToks → generates 10 affiliate videos/day.
   - Tracks ROI via self-hosted analytics (Superset).
3. **Freelancer**:
   - Deploys agents to apply to microjobs (e.g., Fiverr gigs).

Roadmap
1. **Phase 1**: Video auto-editor + basic social posting (6 months).
2. **Phase 2**: Affiliate agents + OAuth security (12 months).
3. **Phase 3**: Microtask marketplace (18 months).

Final Prompt:
"Build same.dev as an open-source **'Canva + Zapier + Hootsuite'** for AI automation. Prioritize privacy (self-hosting), use OSS AI tools, and empower users to own their data. All features must be scriptable via API for developers."

Need:
- Wireframes for the video editor UI?
- RFC for agent interoperability?
- Contributor onboarding guide?

Let's make it viral with #MadeWithSameDev! 🚀

Yes, you can absolutely add features like ACE.ai and **Simulor** into your mgx.dev clone — and here's how you can break it down clearly:
What You Can Add from ACE.ai**:**
ACE is known for **autonomous agents with multiple thinking layers** (aspirational, planning, execution, feedback). You can integrate:
Goal-Oriented Agents (Layered Reasoning)
Memory + Planning Chains
Auto-correct, retry, adapt logic
Skill plugins or modules (e.g. learn how to act like a marketer, narrator, or coder)

How to implement it:
Use 07Builder as your ACE-like master agent
Add planning + execution layers with a LangChain or Continue.dev stack
Integrate memory (vector or local), state tracking, and reflex logic
What You Can Add from Simulor:
Simulor is focused on **interactive NPC-like character simulations**. You can integrate:
Character memory, behavior, and evolving personality
Emotionally reactive dialogue
Scene-based interaction
Voice + animation for avatars (your virtual influencer engine)
How to implement it:
Use your perchance-style storyteller or virtual influencer engine
Add behavior trees or stateful LLM loops (like ReAct or AutoGPT logic)
Integrate with ElevenLabs (voice), AnimateDiff (animation), and Whisper (voice recognition)
Final Prompt for same.dev **(Add ACE + Simulor Style Features):**
Add support for ACE.ai-style multi-layer agent architecture in 07Builder, including planning, reflection, execution, and feedback. Enable dynamic skill-switching and adaptive reasoning. Also, add Simulor-style emotional NPC simulation and interaction engine under the Storyteller or Virtual Influencer module. This should support persistent personality, scene understanding, character memory, voice, and animation. Integrate all into the mgx.dev-style interface with proper section assignment and real-time control from mobile or desktop.

Let me know if you want this fully implemented in agent YAML, UI code, or LangChain config — I'll generate it for you.

Md🆙 UXUY WalidMeshchain.Ai:
Open sidebar

ChatGPT

Of:
can you tell me the detail I wana create ai Agent automatic storytelling videos genaretar based on story with clone and modify voice in n8n tell me. or tell me the ai or model or template name. ensure you need every details I don't have any knowledge about this and free opensource if it's only way for free unlimited run it locally befor I'll try in wab and findaute this is easy to understand
ChatGPT:
Reflecting on AI agents and storytelling videos for 19 seconds
Below is an overview of how you can build an AI-driven automatic storytelling video generator that clones and modifies voices—all using free, open-source components and integrating with n8n for workflow automation. I'll break down the key components, possible models, and a high-level pipeline you can follow.

1. Story Generation
What you need:
- An open-source language model to generate or expand a story script.
- Model options:

– GPT-Neo/GPT-J (by EleutherAI) or BLOOM are good choices because they're free, available locally, and can generate creative narratives.

– Alternatively, you might explore OpenAssistant (an open-source variant) if you need conversational elements.

Implementation tips:
- You can run these models locally using libraries such as Hugging Face's Transformers.
- Create a script that takes a story prompt and generates a full narrative.

2. Voice Cloning and Modification
What you need:
- A text-to-speech (TTS) system that supports voice cloning (taking a sample voice and replicating its characteristics).
- Model options:
  – Real-Time Voice Cloning by Corentin Jemine is a popular open-source project that uses SV2TTS.
  – Coqui TTS or Mozilla TTS are alternatives that also support voice cloning and customization (e.g., pitch, speed).

Implementation tips:
- Train or fine-tune the model with a few voice samples to clone a target voice.
- Use the generated story script as input to produce narration.
- Explore voice modification parameters within these tools to adjust tone, emotion, or accent.

3. Video Creation
What you need:
- A method to generate visual content that fits the narrative.
- Approaches:
  – Text-to-Image models: Use models like Stable Diffusion to generate images or storyboard panels from text prompts.
  – Video synthesis: While full video generation is more experimental, you can combine generated images into a video slideshow with transitions.
  – Additional Tools: For smooth transitions or animated effects, consider using open-source video editing libraries such as MoviePy or command-line tools like ffmpeg.

Implementation tips:
- Define key scenes from the story, generate corresponding images, and then sequence them with timing that matches the narration.
- Overlay the cloned narration audio onto the video timeline.

4. Integration with n8n
What you need:
- n8n is an open-source workflow automation tool that can orchestrate your entire pipeline.
- Integration steps:
  1. Trigger: Set up an n8n workflow that starts when you input a new story prompt or a scheduled event.
  2. Script Nodes: Use custom nodes (or HTTP Request nodes if you wrap your Python scripts in a web API) to call your story generation script.

3. TTS & Voice Cloning: Trigger your TTS/voice cloning script next, passing the generated script.
4. Visual Generation: Trigger the image generation (or video assembly) process.
5. Final Assembly: Use another node to call your video assembly script (using ffmpeg or MoviePy) that combines images and audio into a final video.
6. Output/Delivery: Save the video locally or send it to another service.

Implementation tips:
• Each step can be a separate workflow in n8n, so you can debug and update them independently.
• Use n8n’s file system or HTTP nodes to connect your locally running services.

5. Hardware and Resource Considerations
• Running language models, TTS, and image generation locally can be resource-intensive. Ensure you have a machine (or GPU-enabled server) with sufficient memory and compute power.
• Many of these models support Docker images or virtual environments, which can help streamli

Here’s a **ready-to-use prompt template** for a **10-20 second hook** to grab attention and force viewers to watch the full video:

Prompt Structure
"Create a [10/20]-second video hook with:
1. **First 3 Seconds**: [Shocking stat/question/visual] + [trending sound clip]. Example: *‘What if I told you 90% of people waste money here?’* + intense music.
2. **Next 5 Seconds**: Tease a cliffhanger. *Visual*: Glimpse of transformation (e.g., ‘before’ chaos vs. ‘after’ success). *Text*: *‘This trick saved me $10K…’*
3. **Final 2 Seconds**: Curiosity gap + CTA. *Visual*: Blurred result + arrow pointing to ‘Watch Full Video’. *Voice*: *‘See how in 60 seconds → [Channel Name]!’*

Style: Fast cuts, bold text, high-energy music. **Goal**: Make viewers fear missing out if they scroll."**

Example Prompt
"Generate a 15-second hook for a finance tips video. Start with a shocking text: ‘Stop wasting $500/month!’ with a cash-burning visual and viral sound. Show a blurred before/after budget chart, then flash ‘HOW?’ with a red arrow pointing to ‘Watch Full Video’. Use rapid zooms and trending transitions."

Tools to Execute
- **AI Hook Script**: ChatGPT (add “write a viral hook for [topic]”).

- **Voiceover**: ElevenLabs (energetic AI voice).
- **Visuals**: Canva (templates) + CapCut (dynamic edits).

Need a niche-specific version? 🔥

Could you please provide more details about the desired style, visuals, and overall tone for the video? For example, what kind of animation or live-action footage do you envision? What is the target audience?

To help your Firebase/Dialogflow agent handle video creation requests effectively, follow these steps to structure the necessary parameters and responses:

1. **Define Intent Parameters in Dialogflow**

- In your Dialogflow agent, locate the intent responsible for video creation (e.g., "Create Video").
- **Add required parameters**:
    - style
      (e.g., "cartoon", "minimalist", "3D animation")
    - visuals
      (e.g., "bright colors", "dark theme", "abstract shapes")
    - tone
      (e.g., "playful", "serious", "inspirational")
    - animation_type
      (e.g., "2D", "stop-motion", "CGI")
    - target_audience
      (e.g., "children aged 5-10", "corporate professionals")

- **Mark these parameters as "Required"** so Dialogflow prompts the user if they’re missing.

---

2. **Set Up User Prompts**

For each parameter, add a prompt like:

- *"What style are you imagining for the video? Examples: cartoon, live-action, or 3D animation."*
- *"Should the visuals be bright and playful or dark and moody?"*
- *"What tone are you aiming for? Funny, educational, or something else?"*

This ensures the agent asks clear, specific questions.

---

3. **Provide a Sample User Response**
When testing, reply with a **single message containing all required details**, e.g:
> *"I need a **cartoon-style animation** with **bright colors and playful characters**, targeting **children aged 5-10**. The tone should be **fun and educational**."*

Dialogflow will extract parameters like
style=cartoon
,
target_audience=children
, etc.

---

4. **Update Fulfillment Logic**
In your Firebase Cloud Function (or webhook), ensure the agent processes these parameters. Example:

```
function handleVideoRequest(agent) {
        const style = agent.parameters.style;
        const audience = agent.parameters.target_audience;
        // Use these parameters to generate a response
        agent.add(`Creating a ${style} video for ${audience}...`);
}
```

---

5. **Test and Refine**
- Use the Dialogflow Simulator to test the flow.
- If the agent still asks for missing details, check:
    - Are parameters marked as **Required**?
    - Are prompts defined for each parameter?
    - Are entities (e.g.,
@sys.audience
) correctly mapped?

By structuring your agent this way, it will collect all necessary details upfront and avoid repetitive questions. Let me know if you need help with specific Dialogflow screenshots or Firebase code! 🚀

Below is a comprehensive prompt you can use in Firebase Studio to create your agent. This prompt explains everything—from setting up project files to defining endpoints for multiple video-related functionalities (including converting any video to animation). You can adapt and expand this prompt to fit your project's specific requirements.

Firebase Studio Agent Creation Prompt: VideoMagix

Objective:

Build an AI-powered agent named **VideoMagix** that generates various types of video content, supports multiple multimedia formats, and includes a key feature to convert any video into an animation. This agent should be designed for scalability, easy maintenance, and integration with external video processing APIs.

Project Setup:

Project Initialization:

Create a new Firebase project.

In Firebase Studio, initialize your project with a proper directory structure.

Create key files: a main script (e.g., agent.js or index.js) and a configuration file (e.g., config.json).

Directory/File Structure:

agent.js or index.js: Contains the main logic for handling API endpoints, processing requests, and integrating with video services.

config.json: Manages configuration data such as API keys, default settings, and environmental variables.

/functions **(optional)**: If using Firebase Cloud Functions, include a folder containing individual function scripts for different endpoints.

/utils **or** /services: Contains helper modules for video processing, file management, and conversion logic.

Agent Functionalities:

Video Generation and Styling:

The agent must accept user inputs specifying the desired video style (e.g., cinematic, documentary, animated).

Based on the input, the agent should call a video generation engine or leverage external APIs that can process style-specific video creation.

Integrate parameters for video quality, resolution, audio settings, and optional text overlays.

Multi-Type Content Support:

Enable the agent to manage not only videos but also integrate additional content types (audio, image overlays, textual content).

Structure the endpoints so that each content type is processed or combined as needed.

Video-to-Animation Conversion:

A critical feature of VideoMagix is the ability to convert any video into an animation.

Implement a dedicated endpoint that accepts a source video file, then applies conversion algorithms (for example, leveraging libraries like FFmpeg, custom ML models, or third-party services).

Allow users to specify animation styles (e.g., hand-drawn, cartoonish, or stylized effects).

Implementation Details:

Firebase Cloud Functions (or equivalent backend):

Write endpoints for:

/generateVideo: Process requests for generating videos based on user-defined styles.

/convertToAnimation: Accept a video file upload, process it to extract frames, apply animation transformation filters, and return an animated video.

Use Express (or similar framework) to define and route these endpoints.

File and Data Management:

Use Firebase Storage to manage uploaded video files and store processed output.

Ensure your configuration in config.json includes storage bucket settings, API keys for external video/animation services, and any necessary authentication keys.

Use detailed inline comments within your code to document each section and function.

Error Handling and Scalability:
Implement robust error handling for file uploads, API failures, and conversion errors.
Use asynchronous processing (e.g., promises, async/await) to avoid blocking operations.
Add logging for debugging and monitoring through Firebase’s built-in logging tools.
Testing and Documentation:
Develop unit tests for each endpoint to ensure the functionality of video generation, multi-format content creation, and animation conversion.
Document API endpoints with clear instructions on required parameters, expected results, and error messages.

Consider integrating a real-time preview feature that allows users to see a quick processing result of their request.
Complete Prompt Example:

```
/* * VideoMagix Agent - Firebase Studio Implementation * * Objective: Build an all-in-one
video content generator and converter. * Features: * - Generate videos in multiple styles
(cinematic, documentary, animated, etc.) * - Support for multimedia content: video, images,
text overlays, audio tracks. * - Convert any video to animation using advanced processing
techniques. * * Project Structure: * - agent.js: Main script handling endpoints. * - config.json:
Configuration settings (API keys, default parameters, etc.) * - /functions: (Optional) Cloud
Functions scripts for API endpoints. * - /services: Utilities for video processing and
conversion. * * Endpoints: * 1. POST /generateVideo * - Parameters: { videoStyle, resolution,
audioOptions, textOverlays, ... } * - Description: Processes the input parameters and calls
the video generation engine. * * 2. POST /convertToAnimation * - Parameters: {
sourceVideo, animationStyle, outputFormat, ... } * - Description: Converts an uploaded video
file into an animation based on the specified style. * * Integration: * - Use Firebase Storage
for handling video uploads and processed files. * - Integrate with external video processing
APIs (e.g., FFmpeg, custom ML models). * * Error Handling: * - Robust error handling for file
processing and API failures. * - Use asynchronous processing (async/await) for efficient
handling. * * Testing: * - Implement unit and integration tests for each endpoint. * - Write
clear documentation for API usage. * * Instructions: * - Clone or create the project in
Firebase Studio. * - Set up your configuration in config.json. * - Deploy your endpoints via
Firebase Cloud Functions (if applicable). * - Monitor logs and update configurations as
needed. */ // Example code snippet for /convertToAnimation endpoint in agent.js const
express = require('express'); const app = express(); const bodyParser =
require('body-parser'); app.use(bodyParser.json()); // Dummy function to simulate video to
animation processing. async function convertVideoToAnimation(videoFile, animationStyle) {
// Integrate your processing logic here (e.g., call FFmpeg or a custom API). return { success:
true, animatedVideoURL: 'https://path.to/animated-video.mp4' }; }
app.post('/convertToAnimation', async (req, res) => { try { const { sourceVideo,
animationStyle } = req.body; // Validate input parameters. if (!sourceVideo || !animationStyle)
{ return res.status(400).json({ error: 'Missing required parameters.' }); } const result = await
convertVideoToAnimation(sourceVideo, animationStyle); if (result.success) {
res.status(200).json({ animatedVideoURL: result.animatedVideoURL }); } else {
res.status(500).json({ error: 'Failed to process the video.' }); } } catch (error) {
res.status(500).json({ error: error.message }); } }); // Start server (or export as a Firebase
Cloud Function) const PORT = process.env.PORT || 3000; app.listen(PORT, () =>
console.log(
VideoMagix agent running on port ${PORT}
)); module.exports = app;
```

Notes & Best Practices:

Scalability: As your agent grows, consider splitting functionalities into separate microservices or functions.

Security: Secure API endpoints using Firebase Authentication and validate all inputs to prevent malicious uploads.

Documentation: Keep your code well-documented—this assists in maintenance and lets new developers quickly understand the system.

This detailed prompt should serve as a solid foundation in Firebase Studio for creating your VideoMagix agent. It covers setting up your project, managing files, defining endpoints, and implementing advanced video-to-animation conversions.

Analysis:

I reviewed best practices for Firebase-based AI agents and video processing functionalities from multiple sources.

The provided prompt integrates essential details like project structure, error handling, scalable endpoint development, and third-party API integration—all key elements for building a robust, multi-type video content generator. This approach aligns with current standards for developing cloud-based multimedia generators and provides a detailed blueprint adaptable to your specific needs.

Implement API integrations for video generation and animation using genkit
Implement the selected advanced features for video enhancements. how to do it?

```
# Import necessary libraries
import firebase_admin
from firebase_admin import credentials, firestore

# Initialize Firebase
cred = credentials.Certificate("path/to/your/firebase_credentials.json")
firebase_admin.initialize_app(cred)

# Access Firestore database
db = firestore.client()

# Function to handle user input
def process_input(user_input):
  # Use Dialogflow or other NLU engine to understand intent
  # ...

  # Perform actions based on intent
  # ...

  # Generate response
  # ...

  return response
```

```
# Main loop
while True:
  user_input = input("User: ")
  response = process_input(user_input)
  print("Agent:", response)
```

13 May 2025
02:35

Skip to content
Open sidebar
ChatGPT

You said:
I wnat to make proper explanation list of aboute my agent. like my agents What can do, how can do, what features well have and use and others details. 1# 07builder : run in locally. on my command 07b can create and build completely anything AI agent, web agent, Apps, application, and complex task. If he need to access anything important to complete the task I give aske me. If anything need for complete the task I give like intorogete AI tool or website aske me for APIS and if api is not available aske me for control my mause like ACE.ai. and other feature we dissuaded same
ChatGPT said:
Here's a well-structured explanation list for your agent 07Builder (07b), outlining its purpose, capabilities, features, and usage details. You can use this for documentation, promotion, or onboarding purposes:

Agent Name: 07Builder (07b)

Overview
07Builder, or 07b, is a powerful local AI agent designed to act as your master assistant for automating complex development, creation, and control tasks. It operates entirely on your machine, ensuring speed, control, and privacy.

Core Capabilities
1. Builds Anything on Command
- AI Agents (e.g., assistants, content bots, automation bots)
- Web Agents (crawlers, bots with internet capabilities, etc.)
- Apps (mobile or desktop applications)
- Full Applications (front-end + back-end)
- Complex Tasks & Workflows (AI pipelines, integrations)

2. Task-Aware Development
- Understands tasks described in natural language
- Plans step-by-step how to complete the task
- Executes code, generates files, configures APIs, or integrates tools

3. Permission-Based Execution
- Requests your approval before accessing sensitive tools or data
- Asks for API keys, access credentials, or other required resources
- If no API is available, can ask for mouse/keyboard control (like ACE.ai) to operate manually

Key Features
- Runs Locally: No cloud dependency. Works offline or with selective online access.
- Multi-Agent Creation: Can generate other AI agents and assign them tasks.
- Code + Command Fusion: Can switch between writing code and issuing system-level commands.
- Interrogative Mode: If a tool or API is missing, 07b will ask the right questions to continue.
- Mouse Control Integration: Uses manual control with user permission when automation is blocked (ACE.ai-style).
- Expandable Features: New tools, APIs, models, or workflows can be plugged in anytime.

Example Use Cases
- “07b, build me a Telegram bot that summarizes PDF files.”
- “Create a web scraper agent that checks product prices daily.”
- “Design a full-stack React + Firebase app with user login.”
- “Generate a storytelling AI with voice cloning and image generation.”

User Interaction Flow
1. User gives a high-level command e.g., "Build an app that turns blogs into narrated videos."
2. 07b asks clarifying questions (if needed) e.g., "Do you have a preferred voice clone API?"
3. 07b starts planning and coding
- Searches locally for tools
- Uses available APIs or asks for access
- Builds UI/UX, back-end, logic, and deployment setup
4. Execution & Testing
- Runs everything on your machine
- Shares previews, tests, or results for approval
5. Final Output
- Deliverable (app, agent, system) is generated and optionally saved/exported

Would you like me to turn this into a visual document or slide format for sharing?
You said:
aalso cantrol agent and can easily Integration anything. can help me in vscode. can access and processing audio video, doc, pdf, and file
ChatGPT said:
Great additions! Here’s the updated and expanded explanation list for your 07Builder (07b) agent, now including control features, file processing, and VS Code integration:

Agent Name: 07Builder (07b)

Overview
07Builder, or 07b, is your personal local AI super-agent that can build, control, integrate, and automate anything—from creating other AI/web agents and full applications to controlling your system for task execution. It operates securely on your l

ocal machine, giving you full power with privacy.

Core Capabilities

1. Builds Anything You Describe
- AI Agents (e.g., task bots, conversational agents)
- Web Agents (scrapers, API-based bots)
- Full Applications (desktop, mobile, web)
- Workflows & Automations (e.g., AI video pipelines, chat automation)

2. Self-Expanding Intelligence
- Can build, manage, and deploy other agents or apps
- Learns from tasks to optimize performance

3. Control & Integration
- Can control and communicate with other agents (master-agent mode)
- Easily integrates APIs, tools, and platforms you choose
- Uses system-level access (with permission) for manual control when APIs aren't available

Smart Collaboration with You
- Interrogates for Missing Resources Asks if something is needed to finish a task (e.g., API keys, login, web access).
- Manual Control Option (ACE.ai-style) If automation is blocked, 07b can request permission to use your mouse/keyboard to complete actions like a human.
- Permission-Based Execution You stay in control—07b only performs actions you allow.

Developer Support
- VS Code Integration
- Can assist directly inside your VS Code
- Writes, explains, edits, and runs code
- Sets up local servers, debuggers, and extensions
- Builds Complete Projects in VS Code
- Auto-generates file structures
- Installs dependencies
- Configures build and deployment settings

File & Media Processing

07b can open, read, analyze, and use the following file types:
- Documents: DOCX, PDF, TXT, CSV
- Media: Audio (MP3, WAV), Video (MP4, MOV)
- Others: JSON, XML, Markdown, Code Files (Python, JS, etc.)

Use Cases:
- Summarize or convert PDF to web content
- Transcribe and analyze audio
- Extract highlights from a video
- Convert files into different formats
- Process datasets for AI models

Use Examples
- "07b, create a Django web app that reads PDFs and shows a summary."
- "Build a storytelling AI that processes voice and adds images to make videos."
- "Install and set up a voicebot in VS Code that talks like a movie character."
- "Monitor a folder and notify me when a video is added."

- “Convert this audio to text and generate subtitles.”

Would you like this turned into a one-page PDF, Notion template, or interactive UI layout?
You said:
can you analysis our previous chats if we forget anything or any feature 07builser
ChatGPT said:
Yes, I can absolutely review and analyze our previous chats for anything we might have missed. Here's a feature and function summary of 07Builder (07b) based on everything you've shared so far, including your current goals and vision:

07Builder (07b) – Feature Analysis & Summary
Core Purpose
A local master AI agent that can build other agents, apps, tools, and workflows on command, help you code, automate complex tasks, and control your system with permission.

Confirmed Features & Abilities
1. Build & Automate
- Can build:
- AI agents (specialized or general-purpose)
- Web agents
- Full-stack apps (desktop, mobile, web)
- Custom pipelines (e.g., video/storytelling automation)
- Automation bots (e.g., file watchers, converters)
- Supports both natural language command input and code-level control

2. Developer Toolkit
- Full integration with VS Code:
- Writes, edits, and runs code
- Manages file structures and environments
- Can help with debugging, extension setup, etc.
- Understands and works with:
- Python, JavaScript, HTML/CSS, Node.js, etc.
- Frontend/backend frameworks
- APIs and cloud tools (Firebase, OpenAI, etc.)

3. Control & Expansion
- Acts as a control center for other agents or tools
- Can create and manage sub-agents
- Permission-based access to sensitive tools or commands
- Manual control mode (ACE.ai-style mouse/keyboard control when automation isn’t possible)

4. Communication & Interrogation
- Smartly asks you:
- For clarification on tasks
- For API keys, credentials, tool choices
- Whether to continue with manual actions when APIs fail

5. File & Media Processing
- Can access, read, and process:
- Text files (T

XT, MD)
- Documents (PDF, DOCX, CSV)
- Audio (WAV, MP3)
- Video (MP4, MOV)
- Code files (JSON, XML, PY, JS, etc.)
- Tasks it can perform:
- Summarization
- Transcription
- Format conversion
- Data extraction
- File watching and triggering actions

6. AI Video & Storytelling Integration
- Supports your videoMagix agent ecosystem
- Can:
- Create storytelling workflows
- Assign voice clones to characters
- Handle image, voice, and animation generation
- Convert scripts to narrated video

7. Expandability & Plug-in System
- You can add:
- New models (e.g., Ollama with LLaMA 3)
- New tools (e.g., Firebase, LangChain, Hugging Face)
- New templates or UI modules
- Google Drive/Dropbox integration

Optional Features You Might Still Want to Add
- UI or dashboard for 07b (to control tasks visually)
- Logging and replay system (track what 07b did and repeat tasks)
- Voice command control (speak to 07b)
- Custom personality or memory (optional persistent knowledge or style)
- Multi-user support (collaborate with team members)

Would you like this turned into a structured Markdown readme, visual diagram, or starter config script for your 07b system?
You said:
I Have two question one can I create this AI agent in replit with free and opensource things two can I ad more advance feature free? and I Have the UI interface plan

ChatGPT said:
Great questions. Let's break this down clearly:

1. Can You Build 07Builder on Replit Using Free & Open-Source Tools?
Yes, but with limitations. Here's what you need to know:
Pros (What You Can Do on Replit Free Tier)
- Use Python, Node.js, JavaScript, etc.
- Run code 24/7 (with Replit's free "Always On" limitations)
- Use open-source libraries and packages (via pip, npm)
- Create a basic UI using Flask, FastAPI + HTML/CSS/JS (or Replit's webview)

- Access files, build logic, integrate basic APIs

Cons / Limitations

- Limited RAM and CPU (not ideal for heavy tasks like video processing or LLMs)
- Cannot install custom system-level dependencies easily (e.g., ffmpeg for video/audio)
- No GPU access (so no heavy image generation or local LLMs like LLaMA 3)
- File size limits & sandboxed filesystem
- Limited background processing unless upgraded

Best Use on Replit: Lightweight version of 07b that can code, build agents, run workflows, talk to APIs, and offer a UI.

2. Can You Add Advanced Features for Free?

Yes — many! Here's what you can add with free and open-source tools:

Advanced Free Features You Can Add

Feature Tool/Library Free & Open Source?
Agent building & code writing LangChain + Python Yes
AI text generation OpenRouter (free key) / LM Studio + Ollama Yes
File handling (PDF, audio, etc.) PyMuPDF, moviepy, pydub Yes
Audio transcription Whisper (via Hugging Face or local) Yes
Voice cloning TortoiseTTS, Bark Yes
Image generation Stable Diffusion (via API or Colab) Yes
Video editing moviepy, ffmpeg-python Yes
Manual control (ACE.ai-style) AutoHotkey or PyAutoGUI (on local, not Replit) Yes
UI interface Gradio / Streamlit / Flask + JS Yes

Tip: For GPU/processing-heavy features, host those parts in Google Colab and connect them to Replit via API.

3. UI Interface Plan

If you already have a UI plan, that’s perfect! You can:

- Build it using Gradio (easy + auto-generates interface)
- Or use Flask + HTML/CSS + JS for custom UI
- Or connect to Streamlit for live interface on Replit or locally

Final Suggestion:

- Use Replit to host the core logic, lightweight UI, and agent controller
- Use Colab or local setup (with Ollama, Whisper, ffmpeg) for heavy AI processing
- Use free APIs and open-source tools to bridge features
- You can build a very advanced system without paying, but you’ll need to distribute tasks smartly

Would you like a starter project template or Replit + Colab hybrid setup for your 07b agent?

You said:

I want create it on replit can I? and run it somewhere else's and hosting if needed

ChatGPT said:

Yes, you can absolutely create 07Builder (07b) on Replit, and run or host heavier parts elsewhere. Her e's how to structure it:

How to Create & Run 07Builder on Replit (with External Support)

1. What You Can Build Directly on Replit (Free)

You can build the core of 07b on Replit:

- The main controller agent logic (task parsing, decision-making)
- Agent creation scripts
- Integration with:
- APIs (OpenAI, Hugging Face, Firebase, etc.)
- Tools like LangChain
- UI using Gradio, Flask, or HTML/JS
- Local command interpreter (for creating apps, agents, workflows)
- Simple file parsing (PDF, text, etc.)

2. What You Should Run Elsewhere

Due to Replit's limitations (no GPU, limited RAM/CPU), run these on another platform:

Feature Recommended Hosting Free Option?
LLaMA 3 / Ollama Your own machine / Hugging Face Inference API / Colab Yes
Voice cloning Colab + Bark or TortoiseTTS Yes
Video/image generation Google Colab (with SD, MoviePy, etc.) Yes
Advanced file/audio/video processing Local machine or Colab notebooks Yes
Desktop automation (ACE.ai-like control) Only possible on your own machine (e.g., with PyAutoGUI) Yes

3. How to Link Replit + External Tools

You can create a hybrid system:

- Replit: Runs 07b’s logic, UI, agent builder, and task interpreter
- Colab or Local: Acts as a “processing engine” — triggered via API or webhook

Example Workflow:

1. User gives command in Replit UI: “Turn this story into a narrated video”
2. 07b triggers Colab or local script via webhook or REST API
3. Colab runs Whisper, Bark, SD, etc.
4. Final result is sent back to Replit for display or download

4. Hosting Options

If Replit free tier is too limited, you can later host your 07b app on:

- Render.com (free tier for Python apps with background tasks)
- Railway.app
- Fly.io (good for lightweight backend agents)
- Vercel / Netlify (for front-end and API proxying)
- Your own PC or Raspberry Pi (for max control)

TL;DR – Yes, You Can:

- Build 07b’s core on Replit free
- Run heavy stuff externally (Colab/local/other hosts)
- Host or move later if needed

Would you like a ready-made Replit template for 07Builder to start coding today?
You said:
is ther any way to Integration to replit for unlimited use
ChatGPT said:
There is no way to get truly unlimited use on Replit's free tier, but you can greatly extend its power with smart strategies. Here's how:

How to Maximize or Simulate "Unlimited" Usage on Replit
1. Use External Resources to Offload Heavy Tasks
Offload tasks Replit can't handle well:
Task Host Instead On Integration Method
Video, audio, image processing Google Colab, Hugging Face Spaces, Render REST API, Webhooks
LLaMA/Ollama or Whisper Your own PC with public tunnel (e.g., ngrok) API calls
File storage Google Drive, Dropbox API integration

Result: Replit just becomes a smart control center—lightweight, but powerful.

2. Use External API Credits + Free Tools
Use free-tier APIs that don't cost money:
- OpenRouter (for GPT models)
- Hugging Face (limited free access to models)
- Firebase (Free tier for DB/auth)
- Google Colab (for GPU tasks)
- GitHub + GitHub Actions (to run background scripts)
- Glitch (host microservices for free)

3. Use Tunneling for Local Hosting
Run heavy tasks on your own PC and connect Replit to it via:
- Ngrok (exposes local port as public URL)
- Cloudflare Tunnel (free & more stable)
- Localtunnel

This gives you "unlimited compute" from your PC while using Replit as the UI/interface/brain.

4. Upgrade to Hacker Plan (Optional)
If you're okay spending $7/month, Replit's Hacker Plan gives:
- Private Repls
- Always-on apps (with more limits than true servers)
- More storage and RAM
- Better performance

Still limited, but enough for many AI microservices and development.

5. Turn Replit into an Orchestrator
Design 07Builder to:
- Accept tasks via Replit interface
- Decide where to route it (Replit, Colab, Local, Cloud API)
- Return results to the user via Replit UI

That gives you nearly “unlimited” power while keeping your core system free.

Final Answer:
You can’t get truly unlimited resources on Replit itself for free,

but you can design 07Builder to use Replit as a control center and plug in unlimited compute using:
- Your PC
- Colab
- Free APIs
- Cloud tunnels

Would you like a starter template for a “Replit Orchestrator” setup that connects to external tools like Colab and your local machine?

You said:
If I create in replit i need a prompt give me. if I Use Tunneling for Local Hosting I don’t need replit anymore raithe
ChatGPT said:
Exactly — here's the key:
If You Use Tunneling (Local Hosting via ngrok or Cloudflare Tunnel):
- You don’t need Replit at all unless you want:
- A cloud-accessible UI
- To keep it running when your PC is off
- Your own PC becomes the host for 07Builder
- More powerful because there are no limits — full access to your hardware, files, mouse, tools, etc.
- Perfect if you want to do local control, video/audio processing, and desktop automation (like ACE.ai)

Replit is best when you need a simple online IDE or want to share something easily.
But for serious power, local with tunnel is best.

Here's What You Can Do:
A. If You Choose Replit (Cloud Control Panel)
Use this starter prompt to initialize your 07Builder in Replit:

python
Copy code
```
""" You are 07Builder, a local-first, modular AI control agent.
Your job is to: - Receive commands from the user - Build AI agents, apps, or workflows - Ask for missing info or permissions (API keys, file access, etc.) - Delegate heavy processing to external services (Colab, local tunnels, etc.) - Build custom pipelines using open-source tools (LangChain, Whisper, TTS, video tools)
You can: - Access APIs - Process basic files (text, PDFs) - Generate or modify code - Trigger external endpoints for heavy AI tasks
Always ask before using sensitive data or doing irreversible actions. """
# Starter agent logic (simplified example) import os import requests
```

```
def handle_command(command):     if "create agent" in command.lower():            return
"What kind of agent? Web? Storytelling? Coding assistant?"          elif "process video" in
command.lower():      return "Send the file and I will call the external processor."   else:
          return "Command not recognized yet. Add more capability?"
# Example usage if __name__ == "__main__":        while True:      cmd = input("07b > ")
          if cmd.lower() in ["exit", "quit"]:          break             print(handle_command(cmd))
```

You can build this out with:

- Flask or Gradio for a UI
- Webhooks to your Colab or local tunnel
- LangChain for AI tool orchestration

B. If You Choose Local + Tunnel (No Replit Needed)
Just run your app locally (in VS Code or terminal) and expose it with:

bash
Copy code

```
ngrok http 7860  # or whatever port your UI runs on
```

Then, you can access it from anywhere (including phone, cloud, or Replit frontend if needed).

Summary:

- Replit = cloud frontend + logic, limited compute
- Local + Tunnel = full power, no limits, no Replit needed
- You can combine both if you want control + portability.

Would you like a Colab-compatible processor script, or a local VS Code-based 07Builder setup with ngrok support next?
2/2

You said:
I am asking you I only create my agent in replit then I run it locally. give me the full agent create prompt and tell me how can I run it locally and connect it vscode
ChatGPT said:
Updated saved memory
Got it! You want to create the 07Builder agent in Replit, but then run it locally through VS Code, with full control and unlimited capabilities. Here's exactly how to do that:

Step-by-Step Guide to Build and Run 07Builder

PART 1: Create 07Builder in Replit
A. Agent Prompt (Full Agent Template in Python)
Create a Python file in Replit with this:

python
Copy code

```
# 07Builder Core Agent (Local Control AI Assistant)
```

```
""" 07Builder - Version 1.0 - Modular AI agent creator - Can build apps, workflows, websites,
or agents on command - Can request permissions, APIs, files, or control access - Designed
to run locally or be hosted with external tunnels """
import os
class BuilderAgent:    def __init__(self):       self.na

me = "07Builder"       self.version = "1.0"     self.tools = {             "code_writer": True,
         "api_connector": True,          "file_handler": True,             "tunnel_connector": True
         }
def ask(self, query):             print(f"[{self.name}] Received: {query}")       if "create agent" in
query:            return self.create_agent()      elif "process file" in query:              return
"Please upload the file."        elif "control mouse" in query:           return "Do you grant
permission to control the desktop? (yes/no)"          else:            return "Command not
understood yet. Try 'create agent' or 'process file'."
def create_agent(self):          return (          "What type of agent? (web, AI, voice, video)\n"
         "What tools should I include? (OpenAI, Firebase, LangChain, etc.)"          )
def main():      agent = BuilderAgent()         print("07Builder started. Type a command.")
         while True:      cmd = input(">>> ")             if cmd in ["exit", "quit"]:          break
         print(agent.ask(cmd))
if __name__ == "__main__":  main()
```

PART 2: Export to Local + Run in VS Code

A. Download From Replit

1. Click "⋮" → "Download as ZIP" in Replit
2. Extract it to your local machine

B. Open in VS Code

1. Launch VS Code
2. Open the folder where you extracted the files
3. Ensure you have Python installed (python --version)
4. Create a virtual environment (optional): bash Copy code python -m venv venv source venv/bin/activate # or venv\Scripts\activate on Windows
5. Install any dependencies (if needed)

PART 3: Run It Locally

Just run it:

bash
Copy code

```
python main.py
```

(Replace main.py with whatever filename you used)

PART 4: Optional – Expose It Online (with Tunnel)

Install ngrok or cloudflared and run:

bash
Copy code

```
ngrok http 7860  # if you're serving a web UI later
```

This gives you a public URL to control 07Builder from your phone or any device.

Summary
- Create 07b logic in Replit
- Export it to your machine
- Run and evolve it in VS Code
- Add web UI, integrations, mouse/keyboard control, or tunnels as needed

Would you like a version with Flask or Gradio UI next, so you can control 07b in your browser too?

You said:
one more thing i share you two image tell me create my agent the in fast image and second in vscode can I use their
ChatGPT said:
Absolutely, please go ahead and share the two images. Once I see them, I'll tell you:
1. Whether you can use them to create your agent in the first image (Replit or another interface)
2. Whether the second image (in VS Code) is valid for running or continuing your agent locally
Once you upload, I'll guide you with clear steps based on what's shown in each.
You said:

ChatGPT said:
Thanks for sharing both images! Here's a breakdown of what each one is and how you can use them to create and run your 07Builder agent:

1. First Image (VS Code - AI Agent Environment with Models)
What it shows:
This is the new VS Code AI agent panel with model integration (like Claude, GPT-4.1, DeepSeek, etc.). You also have the option to create your own assistant.
What you can do with this:
- YES, you can use this environment to:
- Create and run 07Builder as a local AI agent
- Access models like Claude 3, GPT-4o, etc. directly in VS Code
- Write code that integrates AI models
- Use the assistant UI to give commands to 07Builder via prompt/chat
How to use it:
- Click "Create Your Own Assistant" → Start building 07Builder
- Connect it with a local Python backend (your logic from Replit)
- You can use Ollama models or connect to OpenAI via API keys

2. Second Image (Create Block UI – Likely from Replit/Teamplan/Custom Assistant Builder)
What it shows:
A block-based assistant builder where you can define:
- Rules
- Prompts
- Data/API access

- Models

What you can do with this:
- YES, you can use this to define your agent structure visually
- Create prompt chains, model connections, and plugin behavior
- Good for a lightweight or web-deployed version of 07Builder

Con

clusion: Can You Use These for 07Builder?
Image Use for Agent Creation Use for Running Locally Notes
1 (VS Code AI UI) YES YES Best for full control and coding your own 07Builder
2 (Block UI) YES (Web UI Agent) Not directly Good for defining logic and deploying it online

Next Step Options
- Do you want me to write the 07Builder assistant config for Image 1 (VS Code Assistant Panel)?
- Or generate a block file YAML/JSON config you can paste into the second platform?

Let me know, and I'll generate it immediately for your preferred setup.

You said:
second image is in continue.dev
ChatGPT said:
Perfect — thanks for confirming! The second image is from Continue.dev — a powerful open-source agent framework that works directly in VS Code and allows you to:
- Create and run custom AI assistants locally or with remote models
- Integrate APIs, files, code tools, prompts, etc. via "Blocks"
- Use models like GPT-4o, Claude, DeepSeek, Phi, etc.
- Seamlessly connect with the Continue VS Code extension

How You Can Use Both Images to Build 07Builder
Image 1: VS Code + Continue Plugin
You already have Continue running with models available. That's your main execution environment.
Image 2: Block Creation UI (Continue.dev dashboard)
You use this to:
- Define 07Builder's features
- Add rules, prompt chains, and APIs
- Make it reusable and publishable
- Choose between Public/Private use

YES, You Can:
- Create 07Builder inside Continue.dev
- Run it fully inside VS Code locally
- Add models, APIs, tools (like file reading, control mouse), etc.
- No need for Replit unless you want a web deployment

Next Steps — Do You Want This?
I can now generate for you:
1. A Continue.dev block config (for second image)
2. A VS Code folder setup that lets you:
- Use 07Builder
- Give it commands like "build X", "control Y", etc.
- Ask for access or permission like you described

Would you like both? Or start with the Continue.dev config block for your assistant?

You said:
If it possible to create 07builder in continue then give me What i need Define 07Builder's features Add rules, prompt chains, and I Have open router, openAI, huggingface token, serp, APIs Make it reusable and publishable

ChatGPT said:
Updated saved memory
Awesome! Here's exactly how to define and build 07Builder inside Continue.dev using your available APIs and capabilities.

07Builder Agent Setup in Continue.dev
1. Define 07Builder’s Core Features
In the "Description" section of the block:

sql
Copy code
```sql
07Builder is a powerful local-first AI assistant that can: - Build AI agents, apps, and web agents from scratch using commands - Interrogate tasks, ask for permission, and request access if needed - Integrate APIs (OpenAI, HuggingFace, SerpAPI, etc.) - Request control over mouse or system (like ACE.ai style) - Help with VS Code tasks, automation, audio/video/doc processing - Modular design: can be extended with new tools, prompts, or models
```

2. Block Fields to Fill in Continue.dev UI

Field	Your Input
Doc Name	07Builder
Start URL	http://localhost:3000 (or leave blank if not needed)
Name	07Builder
Slug	07builder
Description	(Paste the features list above)
Icon	Optional (upload a builder-style icon)
Visibility	Public (so you can reuse/share it)

3. Block Type Selections
Check these:
- Rules – Define when/how 07Builder acts
- Prompts – Base instructions and task prompts
- Blocks – Logic reuse across tools

- Models – GPT-4o, Claude, Hugging Face models
- Data – Docs, PDFs, audio, video
- Context – File/project awareness (for VS Code tasks)

4. Example Rules to Add

yaml
Copy code
```
rules:   - name: Ask Before Acting     content: If any task requires access to APIs, tools, or files, first ask the user for permission.
- name: Agent Construction Rule      content: When asked to build an agent, first ask for type (AI, Web, App), goal, and tools available. Then begin scaffolding.
- name: API Access Rule      content: Use OpenAI, HuggingFace, or SerpAPI as needed. If a required API key is missing, ask the u

ser to provide it.
```

5. Prompt Chains (Main Workflow Prompts)

yaml
Copy code
```
prompts:   - name: Build Agent          content: >        You're 07Builder. The user wants to build a new AI agent. Ask what type (AI, web, app), goal, and toolset. Then begin scaffolding using appropriate tech.
- name: API Integration          content: >        When the user provides an API key or URL, register and test the API. Then show what can be done using it.
- name: Task Assistant          content: >        Help the user with coding, automation, file processing, or integration. If a tool is missing, recommend one.
```

6. Model Connections
In the "Models" section, connect:
- openai/gpt-4o or openrouter/gpt-4.5
- huggingface/mistral or codellama (for coding agents)
- serpapi via API key (for search)
- Local model via Ollama or LM Studio if available

7. Publish and Use
After setting all:
- Click Create Block
- It appears in Continue.dev VS Code panel under "My Blocks"
- You can now invoke 07Builder as your own local-first general agent

Want Me to Generate YAML/JSON Block Config File for You?
I can give you a downloadable config file that you can import into Continue with all this pre-filled. Want that?
You said:
If I give more api or want to interogate coding tool for extend my agent capability
ChatGPT said:

Yes—you can absolutely extend 07Builder anytime by integrating more APIs, coding tools, or even external platforms. Here's how to do it smartly inside Continue.dev:

To Add More APIs or Tools to 07Builder

1. Update Rules
Add new API logic:

yaml
Copy code
```yaml
- name: Use Additional APIs   content: >       If the user provides new APIs (e.g. AssemblyAI, ElevenLabs, Notion, Firebase), store and use them when relevant to the user's request.
```
2. Add Interrogation Prompt

yaml
Copy code
```yaml
- name: Tool Interrogation   content: >          When a task requires capabilities you don't yet have, ask:       "Do you want me to integrate a new tool or API to do this?" Then collect API key or SDK info.
```
3. Extend Prompt Chains for New Tools
Examples:

yaml
Copy code
```yaml
- name: Firebase Integration   content: >       If the user asks for app backend or data sync, suggest Firebase. Ask for credentials and SDK setup. Then initialize a working script or config.
- name: Voice Tool Integration   content: >     For voice features, use ElevenLabs or Bark. Ask for voice clone or TTS mode, then process audio files accordingly.
```
4. Coding Tool Awareness
You can let 07Builder “interrogate” tools like:
- LangChain
- Gradio
- FastAPI
- Tauri
- PyScript
- WebAssembly
- Docker

Just define a prompt like:

yaml
Copy code
```yaml
- name: Tech Stack Advisor   content: >        For any user-requested build, suggest the most suitable coding tools (e.g. FastAPI for backend, Gradio for UI). Ask if user wants to use them, then scaffold the base code.
```

In Practice

When 07Builder hits a limit (e.g. "I can't process audio"), it’ll ask you:

“Should I integrate Whisper or AssemblyAI for this? If yes, please provide API credentials or permission to install it.”

You Can Keep Expanding by:
- Adding new prompt chains
- Saving API keys securely (or prompt user to re-enter)
- Linking Continue.dev to local tools like Ollama, LangChain, TTS engines
- Mounting agent tools as reusable blocks

Want a template to auto-load API integration blocks into your agent?

You said:
I am trying tell you i want to make my agent I ad mani API and interogete mani tool as I can and my agent use based on input or what task I give. Do get it
ChatGPT said:
Yes, I get it now—you want to build 07Builder as a smart, modular AI agent that:
- Can accept many APIs or coding tools from you.
- Can interrogate those tools dynamically.
- Can decide what to use based on the task or input you give it.
- Can ask you for missing info (like API keys or access) when needed.
- Can run locally or in environments like Replit or Continue.dev.

Here's What You’re Building:
07Builder = a self-expanding, command-driven AI engineer that:
- Builds other agents, apps, scripts, or pipelines.
- Has

dynamic tool access (APIs, AI models, coding tools).
- Is aware of when and how to use what you’ve given it.
- Uses intelligent prompting, rule-chains, and memory (if possible).

Example Behavior You Want:
You: “07b, create a story-to-video generator with voice cloning.”
07Builder:
1. Checks if it has access to image + voice APIs.
2. If not: “Do you want to use D-ID and ElevenLabs? Please provide API keys.”
3. Builds and connects them via Python or Gradio.
4. Then confirms: “Toolchain is ready. Want to save this as a reusable agent?”

This is absolutely possible with:
- Continue.dev + prompt blocks
- LangChain or custom Python agent
- API plugin/loader system
- Local tools like Ollama, Whisper, TTS, etc.

Would you like me to now write a complete Continue.dev agent block setup that:
- Accepts any API/tool
- Interrogates based on task

- Dynamically chooses tools
- Asks for what’s missing

?
You said:
yes but I share two file one is 07b file and two is UI interface. can I use this UI interface in continue
ChatGPT said:
Great — yes, please share both files:
1. The 07Builder file (your logic or config)
2. The UI interface file (how you want the interface to look)

Then I’ll tell you:
- If the UI interface can be used directly inside Continue.dev
- Or how you can connect it to 07Builder via Continue or externally
- And how to link that UI to your local or hosted agent

Please go ahead and upload both files.

You need GPT-4o to continue this chat because there's an attachment. Your limit resets after 6:27 AM.
New chat
Get Plus

ChatGPT can make mistakes. Check important info.

Get Microsoft OneNote: https://aka.ms/GetOneNoteMobile

I tell you guide like I am 5 year old what to do wayr to do how to do like aftar create and seve the file go x, then cheek y, then cheek z.