Md🆙 UXUY WalidMeshchain.Ai:

Here is your **complete, professional prompt** that you can use in Continue.dev, **VS Code**, or any free open-source setup (like Replit or a local Python/Node.js project). It covers every detail of your project clearly and deeply — ready to copy-paste:

Project Setup Prompt: "My Personal Manager" (Open-Source mgx.dev **Clone)**

Create a free and open-source clone of mgx.dev called **"My Personal Manager"**. This is a local-first, modular, personal AI workspace that runs fully on desktop (VS Code, Replit, or Continue.dev) with these exact requirements: ### Workspace Identity: - Project Name: My Personal Manager - System / Framework Name: "tames workers" (this is how agents/tools/processes are labeled) - Do not rely on any paid APIs by default. All tools should work with free tiers or offline/local open-source models. --- ### Sections to Include: 1. **Portfolio** - Recreate **perchance.org** features - Add custom storytelling logic - Add virtual influencer base - Use for creative writing, scene logic, personality design 2. **Tools** - General tools like video converter, transcription, video cutter, etc. 3. **Templates** - Templates for story to video, poetry to video, meme to video, tutorial to video, etc. 4. **Agents** Include 4 prebuilt agents here (these are not the main agent): - **Agent 1:** NSFW/Uncensored Creator - **Agent 2:** Multi-language Dubbing Agent (supports anime, movies, etc.) - **Agent 3:** Reels/Short Video Generator (analyzes YouTube or long video links, generates 3 titles, thumbnails, and 3 reels) - **Agent 4:** Interrogator Agent (can analyze and extract functions from tools described in videos and create new versions) 5. **Add Interactive Features** - Keyboard autocorrect, typing enhancement - Voice input to video questions - Real-time dialogue engine 6. **Shortcuts** - Pre-built workflows for video editing, character creation, dubbing, storytelling 7. **Virtual Influencer** - Avatar with persistent voice, emotion, and memory - Social media-ready character builder - Can dub and post content automatically --- ### Main AI Agent: - Name: **07Builder** (replaces "Alax" in mgx.dev) - This is the central autonomous agent like ACE.ai - Supports: - Planning + Execution + Feedback loops - Secure memory and state - Multi-agent controller for Tools/Agents/Templates - Real-time decisions (where to add what based on user prompts) --- ### Additional Features: - Add API + Secrets Manager: - A simple interface with a [+] button - Allows adding API key name + key + save - Easy for future access and auto integration - Add Mobile QR Connect: - Generate QR to sync local app with Android - Access interface from mobile device via local network - Storytelling Styles: - Analyze video input to match storytelling tone - Let user choose "style" (e.g., dark, romantic, anime, cinematic, comedy) - Style & Edit Engine: - Auto-edit videos based on sample style - Add filters, effects, transitions - Suggest music and pacing per genre --- ### Technical Requirements: - Use Python + Node.js stack (or Gradio + LangChain + Continue.dev as needed) - Use open-source models for LLM (e.g., Ollama, Mistral, LLaMA 3 local) - Use Whisper for speech, AnimateDiff or OpenPose for animation, ElevenLabs (optional) for voice --- ### Build Instructions: - Start with UI structure (like mgx.dev) - Create folders and routers per section - Integrate agent loader (for tames workers) - Don't prefill content — just set up skeleton, loading logic, and UI design - All files should be editable and open-source friendly --- This prompt defines everything needed to build **My Personal Manager** without using same.dev, without tokens or paid APIs. Build it offline, run it locally, and control everything from one place.

Would you like me to also:

Generate the base **file/folder structure**?

Write the UI code starter (React + Tailwind or HTML)?

Give you a main.py or LangChain chain loader?
Let me help you now, step by step. You’re **almost there.**

Below is a **step-by-step implementation guide** that pairs with your master prompt. Follow each phase in order—your work will stay perfectly in sync as you build out **My Personal Manager** (the mgx.dev clone) locally and open-source.

Phase 0: Prepare Your Environment

Choose Your Tech Stack

Backend: Python (FastAPI or Flask) + Node.js (Express) or pure Python

Frontend: React + Tailwind CSS (or plain HTML/CSS if preferred)

Agents & LLMs: Local LLaMA 3 / Ollama, Whisper.cpp, AnimateDiff, etc.

Initialize Repository

```
mkdir my-personal-manager cd my-personal-manager git init
```

Create Virtual Environment & Install Dependencies

```
python -m venv venv && source venv/bin/activate pip install fastapi uvicorn openai-whisper llama-cpp-python npm init -y && npm install react react-dom tailwindcss
```

Phase 1: Scaffold Your UI Structure

Create Frontend Folders

```
src/ components/ pages/ styles/
```

Copy mgx.dev **Layout**

Reproduce header, sidebar, footer

Use Tailwind to match typography and spacing

Define Routes for Each Section

/portfolio, /tools, /templates, /agents, /interactive, /shortcuts, /influencer

Implement a Simple NavBar & Sidebar

Links to each section

Mobile-responsive hamburger menu

Phase 2: Build Empty Section “Skeletons”

For each route, create a placeholder page:

Portfolio

```
function Portfolio() { return <div>Portfolio Section (Perchance clone UI here)</div> }
```

Tools, **Templates**, **Agents**, **Interactive Features**, **Shortcuts**, **Virtual Influencer**

Duplicate above with correct titles

Wire Up Navigation

Ensure clicking sidebar items loads the right component

Phase 3: Integrate Main Agent Loader (07Builder)

Backend Endpoint /api/agent/07builder

Loads and routes requests to sub-agents

Front-End Service

Create an AgentService that calls /api/agent/07builder

UI Panel (e.g. in Tools or a floating widget)

Text input + “Send to 07Builder” button

Phase 4: Plug in Your Prebuilt Agents

Under **Agents Section**:

Agent 1: NSFW Creator
Agent 2: Dubbing Agent
Agent 3: Reels Generator
Agent 4: Interrogator
For each, implement:
A card with “Launch” button
Front-end call to /api/agent/{agentId}
Display results in a modal
Phase 5: Portfolio Section – Perchance.org **Clone**
Copy Perchance UI Patterns (lists of generated characters)
Add “Create New Character” Button
Opens a form with fields for logic rules, behavior, emotion tags
Implement Save & Load
Store in local JSON or small database (SQLite)
Phase 6: Tools & Templates
Tools Section
Build UI cards for utilities (transcribe, cut video, style transfer)
Templates Section
List prompt templates (JSON file)
“Use Template” dropdown that pre-fills 07Builder input
Phase 7: Interactive Features & Shortcuts
Keyboard Assistant
Add a global key listener on textareas
Call a “/api/util/grammar” endpoint for suggestions
Voice & Video Input
Integrate navigator.mediaDevices for recording
POST audio/video to /api/util/whisper
Phase 8: Virtual Influencer Module
Avatar Builder UI
Upload or generate images (AnimateDiff)
Assign voice sample
Persona Profile
Name, backstory, traits
Save to /api/influencer/characters
Publish Controls
Buttons to export video or schedule posts (future)
Phase 9: API & Secrets Manager
New Section /settings/apis
“+ Add API” Form
Fields: Name, Key (masked), Test toggle
Encrypt & Store
Use Fernet/AES to encrypt keys in a local file
List & Toggle
Show status, enable/disable
Phase 10: Mobile QR Connect
Generate QR on desktop:
import QRCode from 'qrcode.react'; <QRCode value={sessionToken} />

Backend: /api/session/token
Mobile View
Simple PWA at /mobile that reads token from URL
Phase 11: Storytelling Style & Auto-Edit Engine
Style Selector UI
Dropdown in Portfolio or Tools
Options: Dark, Anime, Romantic, Cinematic, NSFW
Call Style API
/api/util/styleTransfer with params
Preview & Apply
Phase 12: Testing & Synchronization
Run Backend & Frontend

```
uvicorn main:app --reload & npm start
```

Test Each Section
Navigate routes, launch agents, add API keys, scan QR
Fix Layout Breaks
Check mobile view, Real-time sync
Next Steps
After verifying skeletons, begin wiring logic in each section
Incrementally test each feature with sample data
Keep commits small and descriptive
With this guide in hand, you'll move from **blank clone** to **fully functional My Personal Manager**—all in perfect sync and organization. Let me know when you hit each milestone, and I'll help you refine the next phase!

https://drive.google.com/drive/folders/1-IX2UbibejjaGcSMHO6LMXiK-V8bcOS4?usp=sharing

yes I well like to add
Bonus Features You Can Add Next
You already have: #Modular agents, #API key manager, #Clean UI,
#Virtual Influencer Builder — based on perchance logic
#Auto project save/load but different specs
#Voice cloning with different character assignment. also i want add a option for save different character/samples of voice / discription aboute/ Male/female. this option well help better content. any Agent can use thous characters based on content needs.
Chatgpt if you think you well enhance this feature you can do it.
#Agent logs panel (task history, feedback)
#add extra storege like API feature.
and I want to edit or costamizebol option in all agents like system prompt, rule and wayar stor this agent file, optional what to save (categorize) and if you think to add more things optional.

and this is my feavreate the workspace for this i well give you a image this is most important for me I want to same to same workspace UI interface, backend, how it's work. in mgx.dev alax have a tame on this based in my parsnale menegar main agent 07builder also Have a tame and same abilities

but 07builder have same extra feature and ability. features: like ACE.ai and simulor.
and 07 can use any AI, AI tool, website, agents any feature.
I Have a daute I don’t see the chat interface if you need image for this Ask me.
ones we finish to create with all things then we discuss same more clones like perchance.org, invideo.io and filmmora

Great — your vision is very clear, and I fully understand how important the **workspace and tame interface** are to your project. Based on everything you've described, here's the updated blueprint and checklist for implementing your **My Personal Manager** system with all the advanced features, with **07Builder** as the central ACE-style agent.

My Personal Manager: Next Phase Implementation Plan

(Includes the workspace, tame interface, character voices, agent logs, auto save/load, API storage, and full agent control)

1. Virtual Influencer Builder (Perchance-based)

Add logic system like:
Name, age, traits
Emotion presets
Story behavior templates
Mood/scene/dialogue dynamics
UI builder with live preview
Export format: JSON or Perchance syntax

2. Auto Project Save/Load With Specs

Let users create, name, and load:
Story projects
AI agent chains
Settings (voice, character, language, etc.)
Autosave using localStorage or custom backend

```
{ "project_name": "DarkForest Story", "agents": ["StoryBuilder", "VoiceCloner"], "settings": {
"language": "en", "voice": "Tessa (F)", "character": "Sorcerer" } }
```

3. Voice Cloning + Character Assignment System

Data per character:

```
{ "name": "Lara", "voice_sample": "lara_sample.mp3", "gender": "Female", "description":
"Mystical warrior, calm tone", "tags": ["fantasy", "soft", "female"] }
```

UI:

Add/Preview Voice Sample
Assign gender, tags
Save to /characters/ folder
Any agent can call: getCharacter("Lara") and use voice with matched tone

Extra:

Support for ElevenLabs, Bark, OpenVoice, etc.
Local TTS fallback option (Tortoise, Coqui)

4. Agent Logs Panel (Task History + Feedback)

```
[ { "timestamp": "2025-05-23 10:31", "agent": "NSFW Creator", "input": "Scene: steamy
beach", "result": "Generated 3 prompts + 2 images", "feedback": "Too mild, retry" } ]
```

UI Features:

Log filtering by agent
Quick retry or replay
Exportable .txt/.json

5. Extra Storage API Features

Upload/download character sets, logs, voices
Google Drive/Dropbox integration
Add save slots (e.g., slot1.json, slot2.json, etc.)

6. Editable Agent Options UI

For every agent:
System Prompt
Rules
File Path or Save Location
What to Save (checkboxes: voice, settings, log, output, etc.)

{ "name": "07Builder", "system_prompt": "You are an autonomous AI agent with full tool access...", "rules": ["Always confirm before deleting files", "Auto-log tasks"], "save_path": "/agents/07Builder/", "save_options": ["logs", "voice_outputs", "feedback"] }

7. Keyboard Agent Clarification

If you're unsure, here’s a proposal:
It monitors keyboard typing
Offers real-time grammar/logic fixes (like Grammarly + Copilot)
Works as a system tray assistant
Can convert drafts to structured prompts
You can rename this to "Text Assistant" if it fits better.

8. Workspace — The Most Important Part

We’ll recreate the mgx.dev **style workspace**, especially the **Alax’s Tame panel**, in your system for 07Builder.

You said:

“This is most important for me I want the same to same workspace UI interface, backend, how it works.”

Yes — absolutely. I will match:
Sidebar layout
Tame tab
Agent tree
Session pane
Log window
Command bar / prompt bar
Task queue / active agent tracker
Special Features in 07Builder
Your agent is **better than Alax** with:
ACE.ai **engine**
Simulators
Universal AI/Tool Access (any model, website, API, voice, or character)
Next Step: Please Upload
The image of the **exact workspace (**mgx.dev **tame view)** you want to replicate
(Optional) Your rough workspace draft or HTML version if you’ve started it
Then I’ll give you:

React/Vue code template for workspace
Backend logic to match tame behavior
Full layout recreation

Later Goals (Post Workspace)

Once all above is implemented, we'll clone:
Perchance.org (interactive prompt logic system)
invideo.io (script to video engine)
Filmora/Capcut (AI video editor/animator)

Let's begin by building the workspace. Please upload the image, and I'll recreate it **exactly as you want**.

new write a mastar prompt for hol project revision this prompt well be your bast of bast
one more thing write it in 3 part and i am telling Again chat interface and workspaces interface
well be exactly same mgx.dev and 07builder well accept any kind of files code audio video + etc

Below is a **three-part master prompt** you can feed into your local agent framework (e.g. Continue.dev, LangChain, or your own 07Builder) to **revise and finish your entire "My Personal Manager" project**. It ensures:

Exact mgx.dev–style chat & workspace UI
Full support for any file type (code, audio, video, etc.)
Integration of all features we've discussed

Part 1: Contextual Analysis & Completion

You are 07Builder, the core autonomous agent of "My Personal Manager." I've already built a working skeleton with these sections: Portfolio, Tools, Templates, Agents (prebuilt NSFW, Dubbing, Reels, Interrogator), Interactive Features, Shortcuts, Virtual Influencer, API & Secrets Manager, and QR mobile link. Your first task is to: 1. Deeply analyze the existing project files and UI components. 2. Identify missing or broken pieces without overwriting any code I've already written. 3. Auto-complete each section according to my vision, linking agents to the right tools, templates, and UI screens. 4. Ensure every feature (story styles, voice cloning profiles, agent logs, API manager, mobile sync) is fully wired up and functional. 5. Prepare a comprehensive change log summarizing what was added or fixed.

Part 2: Exact mgx.dev **Chat & Workspace UI Replication**

Next, recreate the **chat interface** and **workspace/tame panel** to match mgx.dev exactly: - **Chat Bar** at top: "Ask 07Builder…" placeholder, mic button, QR-scan button - **Sidebar** with icons and hover labels for each section - **Tame Panel** inside Agents view: shows active agent, task queue, logs, and controls - **Dashboard Widgets**: cards for each prebuilt agent with "Activate" buttons, API manager table, Virtual Influencer preview - **Responsive layout** that works identically on desktop and mobile - **Animations, colors, typography, spacing** must follow mgx.dev styles Do not alter the content—only replicate the structure, styling, and interactive behavior so a user can't tell the difference.

Part 3: Universal File & Media Handling

Finally, extend 07Builder's capabilities so it accepts and processes **any file type**: - When a user drags or selects a file (code snippet, audio clip, video, image, text document, spreadsheet, PDF, ZIP), 07Builder should: 1. Detect file type automatically. 2. Route to the appropriate sub-agent or utility (e.g., code agent, transcription agent, video editor, summarizer). 3. Process or transform the file (e.g., run code linting, transcribe audio, generate reels from video, analyze PDF content). 4. Return results inline in the chat panel or open the proper UI module. Ensure all uploads, processing steps, and downloads happen seamlessly within the mgx.dev-style workspace, preserving user context and history.

Paste these three parts into your project's master-prompt field (or feed them sequentially) and let 07Builder **auto-complete, polish, and wire** every piece—delivering your full “My Personal Manager” experience exactly as envisioned.

gh repo clone MGX-metagenomics/MGX-gui

it's not work 😮‍💨 Let's start a new project. fast deeply analysis mgx.dev every single detail accept mgx.dev workspace and Mike Team Leader, Alex Engineer, Emma prodact manager, Bob Architect, David Data analyst. and most
important UI interface.

https://lovable.dev/projects/d56d67db-1c50-48f2-960c-1aa41fc88736

prompt

https://lovable.dev/projects/fafe259a-ff46-45f0-812e-1c598bf4b505

home page

https://lovable.dev/projects/513db1a2-0fcc-4643-bd43-f10d076dfa80

extra featuers

it's not work 😮‍💨 Let's start a new project. fast deeply analysis mgx.dev every single detail accept mgx.dev workspace and Mike Team Leader, Alex Engineer, Emma product manager, Bob Architect, David Data analyst. and ability's and most
important UI interface. i added my old prompt. in old prompt replace mike with my 07builder but new i don't want to replace mike. what i want just ad more features. i explain all what i ad and edit. step 1: clone mgx.dev website if need any file no problem i already download (MGX-gui-MGX_gui_2_0.zip) from GitHub. next step 2: ad features in my prompt. the purpose of create this project only myself and i want from this project help me with my work and run my social media platforms and create multitype content and give my idea and imagination give visualized look.

Thank you for the detailed clarification. Based on your instructions, here's **Part 1** of your master prompt to initiate the project in Lovable.dev:

## **Part 1: Project Initialization and UI Setup**

Project Title: *My Personal Manager*

Objective: Develop a personalized AI-powered workspace by cloning the mgx.dev interface, integrating additional features, and customizing the UI to suit personal preferences.

Instructions:

1. **Clone** mgx.dev **Interface**:

* Utilize the provided MGX-gui-MGX_gui_2_0.zip file to replicate the mgx.dev website structure and functionality.
* Ensure that the cloned interface includes all original components, such as the chat interface, workspace, and team structure (Mike, Alex, Emma, Bob, David).

2. **Integrate Additional Features**:

* Remix and incorporate the following Lovable.dev projects into the cloned mgx.dev interface:

    * Project 1: https://lovable.dev/projects/d56d67db-1c50-48f2-960c-1aa41fc88736
    * Project 2: https://lovable.dev/projects/513db1a2-0fcc-4643-bd43-f10d076dfa80

3. **Customize UI Theme**:

* Modify the color scheme of the cloned interface to a blue theme, reflecting personal aesthetic preferences.

4. **Define 07Builder Role**:

* Introduce "07Builder" as the manager within the system.
* Assign 07Builder the responsibilities of organizing tasks, managing the site using ACE.ai and Simulor capabilities, and having exclusive access to the user's PC.
* Ensure that 07Builder does not interfere with the existing mgx.dev team roles.

5. **Set Up Homepage**:

* Design the homepage to feature two primary options with prominent icons: "Image" and "Storytelling".
* Under the "Image" section, integrate the cloned mgx.dev interface.

6. **Incorporate Essential Sections**:

* Ensure the following sections are present and functional within the interface:

    * Portfolio

   * Tools
   * Templates
   * Agents
   * Interactive Features
   * Shortcuts
   * Virtual Influencer

Note: This project is intended for personal use, aiming to assist in managing work tasks, running social media platforms, and creating diverse content.

Please proceed with this setup in Lovable.dev. Once completed, we can move on to Part 2, which will cover the integration of advanced features and functionalities.

Thank you for your detailed instructions. Based on your requirements, here's **Part 2** of your master prompt to continue building your "My Personal Manager" project in Lovable.dev:

## **Part 2: Advanced Feature Integration and Customization**

Objective: Enhance the cloned mgx.dev interface with additional functionalities, integrate specific Lovable.dev projects, and customize the user interface to align with personal preferences.

Instructions:

1. **Integrate Additional** Lovable.dev **Projects**:

 * Remix and incorporate the following projects into the existing cloned mgx.dev interface:

   * Project 1: https://lovable.dev/projects/d56d67db-1c50-48f2-960c-1aa41fc88736
   * Project 2: https://lovable.dev/projects/513db1a2-0fcc-4643-bd43-f10d076dfa80
 * Ensure seamless integration of features from these projects into the main interface.

2. **Enhance 07Builder's Capabilities**:

 * Assign 07Builder the role of a manager with the following responsibilities:

   * Organize and assign tasks to the mgx.dev team members (Mike, Alex, Emma, Bob, David).
   * Utilize ACE.ai and Simulor capabilities to manage site operations.
   * Maintain exclusive access to the user's PC for managing personal files and data.

3. **Implement Advanced Functionalities**:

   * Develop a Virtual Influencer Builder based on perchance logic, allowing the creation of AI-driven personas.
   * Integrate voice cloning features with options to:

     * Save different character samples.
     * Assign descriptions (e.g., male/female).
     * Enable agents to utilize these voices based on content requirements.
   * Establish an Agent Logs Panel to track task history and feedback.
   * Incorporate an API Key Manager for secure handling of API credentials.
   * Add extra storage capabilities to support API features.
   * Provide options to edit or customize agents, including system prompts, rules, storage locations, and categorization preferences.

4. **Customize User Interface**:

   * Modify the color scheme of the interface to a blue theme, reflecting personal aesthetic preferences.
   * Ensure that the homepage features two primary options with prominent icons: "Image" and "Storytelling".
   * Under the "Image" section, integrate the cloned mgx.dev interface.

5. **Ensure Seamless User Experience**:

   * Maintain a responsive layout that functions identically on both desktop and mobile devices.
   * Ensure that all features and integrations operate smoothly within the interface.
   * Preserve user context and history throughout interactions.([Product Hunt][1])

Note: This project is intended for personal use, aiming to assist in managing work tasks, running social media platforms, and creating diverse content.

Please proceed with implementing these enhancements in Lovable.dev. Once completed, we can move on to Part 3, which will cover deployment and testing procedures.

[1]: https://www.producthunt.com/products/metagpt-x?utm_source=chatgpt.com "MGX (MetaGPT X) Product Information and Latest Updates (2025)"

Thank you for the detailed information. Based on your requirements, here's **Part 3** of your master prompt to continue building your "My Personal Manager" project in Lovable.dev:

## **Part 3: Deployment and Testing Procedures**

Objective: Finalize the project by deploying it on Lovable.dev and conducting thorough testing to ensure all features function as intended.

Instructions:

1. **Finalize Project Setup**:

   * Ensure all previously integrated features, including the cloned mgx.dev interface, additional Lovable.dev projects, and customized UI elements, are properly configured and functional.
   * Verify that 07Builder is set up with the designated managerial role, overseeing tasks without interfering with the existing mgx.dev team roles.

2. **Connect to GitHub Repository**:

   * Integrate the project with a GitHub repository to enable version control and collaboration.
   * Follow Lovable.dev's GitHub integration guidelines to link the project:

        * Navigate to the project settings in Lovable.dev.
        * Select the option to connect to GitHub and authorize access.
        * Create a new repository or link to an existing one as appropriate.

3. **Deploy the Project**:

   * Once the project is finalized and connected to GitHub, proceed to deploy it:

        * Click the "Publish" button located in the top-right corner of the Lovable.dev editor.
        * Follow the prompts to complete the deployment process.
        * Upon successful deployment, a shareable URL will be generated for the live application.

4. **Conduct Thorough Testing**:

   * Access the deployed application via the provided URL.
   * Test all integrated features to ensure they function as intended:

        * Verify the responsiveness and layout of the cloned mgx.dev interface.
        * Test the functionalities of the integrated Lovable.dev projects.
        * Ensure that 07Builder effectively manages tasks without interfering with the mgx.dev team roles.
        * Check the performance of the Virtual Influencer Builder, voice cloning features, Agent Logs Panel, API Key Manager, and additional storage capabilities.
        * Confirm that the customized UI elements, including the blue theme and homepage options, are displayed correctly.

5. **Iterate Based on Feedback**:

* Gather feedback from personal use and identify any areas for improvement.
* Make necessary adjustments within Lovable.dev, utilizing the platform's AI-assisted code generation and editing features.
* Re-deploy the application after implementing changes to ensure updates are reflected in the live version.([Lovable Documentation][1], [Lovable Documentation][2])

Note: This project is intended for personal use, aiming to assist in managing work tasks, running social media platforms, and creating diverse content.

Please proceed with these steps to deploy and test your application on Lovable.dev. If you require further assistance or encounter any issues during this process, feel free to ask.

[1]: https://docs.lovable.dev/faq?utm_source=chatgpt.com "FAQ - Lovable Documentation"
[2]: https://docs.lovable.dev/features/deploy?utm_source=chatgpt.com "Deploy - Lovable Documentation"

Thank you for your patience. Building upon our previous discussions, here's **Part 4** of your master prompt, focusing on advanced integrations and enhancements for your "My Personal Manager" project on Lovable.dev:

Part 4: Advanced Integrations and Enhancements

Objective: Augment the existing project with advanced integrations, AI capabilities, and enhanced functionalities to create a robust, personalized AI workspace.

1. AI Integrations

OpenAI Integration:

Incorporate OpenAI's language models to enable advanced text generation, summarization, and conversational capabilities within your agents.

Utilize OpenAI's API to power features like content creation, idea generation, and automated responses.

Anthropic's Claude Integration:

Integrate Claude models for complex reasoning and deep analysis tasks, enhancing the decision-making abilities of your agents.

Leverage Claude's strengths in understanding nuanced prompts and generating detailed outputs.(Lovable Documentation)

Groq Integration:

Implement Groq's high-speed AI models to facilitate real-time interactions and rapid content generation.

Use Groq for tasks requiring quick turnaround times, such as live chat responses or instant content drafts.(Lovable Documentation)

Reference: Lovable.dev supports these integrations, allowing seamless incorporation of AI services into your application.

2. Backend and Database Enhancements

Supabase Integration:

Connect your project to Supabase to manage authentication, database storage, and real-time data synchronization.

Utilize Supabase's Edge Functions to handle backend logic and API interactions securely.(Lovable Documentation, Lovable Documentation)

Aiven Integration:
Integrate Aiven's managed PostgreSQL services for scalable and reliable database solutions.
Use Aiven to handle large datasets, ensuring efficient data storage and retrieval for your application's needs.(Aiven)
Reference: Combining Supabase and Aiven provides a robust backend infrastructure, enhancing your application's performance and scalability. (Aiven)

3. Enhanced Agent Capabilities

Agent Customization:
Allow users to edit and customize agent parameters, including system prompts, operational rules, and storage preferences.
Implement a user-friendly interface for managing agent configurations, enabling tailored experiences for different tasks.(Aiven)
Agent Logs Panel:
Develop a comprehensive panel to track agent activities, including task history, feedback, and performance metrics.
Use this data to refine agent behaviors and improve overall efficiency.
Voice Cloning and Character Assignment:
Integrate voice cloning technologies to create diverse voice profiles for agents.
Allow users to assign specific voices to agents based on content requirements, enhancing personalization.

4. User Interface and Experience

Blue Theme Customization:
Modify the application's color scheme to a blue theme, aligning with personal aesthetic preferences.
Ensure consistency across all UI components for a cohesive look and feel.
Homepage Configuration:
Design the homepage to feature two primary options with prominent icons: "Image" and "Storytelling".
Under the "Image" section, integrate the cloned mgx.dev interface, providing users with a familiar workspace.

5. Deployment and Testing

GitHub Integration:
Connect your project to GitHub for version control, collaboration, and code management.
Utilize GitHub's features to track changes, manage issues, and collaborate with contributors.(Lovable Documentation)
Deployment via Lovable.dev:
Use Lovable.dev's deployment tools to publish your application, making it accessible via a shareable URL.
Conduct thorough testing to ensure all features function as intended, addressing any issues promptly.(Lovable Documentation)
Reference: Lovable.dev provides comprehensive documentation on deploying applications and integrating with GitHub. (Lovable Documentation)

Note: This project is intended for personal use, aiming to assist in managing work tasks, running social media platforms, and creating diverse content.

Please proceed with implementing these advanced integrations and enhancements in your Lovable.dev project. If you require further assistance or have additional features in mind, feel free to ask.

Nein, das ist KEIN fertiger Prompt für Lovable.dev**.**

Was ich geschrieben habe ist eine **Analyse und Technologie-Übersicht**. Lovable.dev braucht einen **spezifischen Entwicklungs-Prompt**.

## Hier ist der echte Prompt für Lovable.dev:

# My Personal Manager - AI Workspace (Free/Open Source)

Create a desktop AI workspace application using Electron + React with the following structure:

## Main Interface
- **4-section layout**: Portfolio, Tools, Templates, Agents
- **Left sidebar navigation** with icons for each section
- **Main workspace area** with tabbed interface
- **Right panel** for agent controls and logs
- **Dark theme** with modern UI design

## Core Features to Implement

### 1. File Upload System
- Drag & drop zone for videos, audio, PDFs, docs
- File preview and metadata display
- Local storage management with SQLite

### 2. 07Builder Main Agent
- Chat interface similar to ChatGPT
- Real-time typing indicator
- Message history with SQLite storage
- File attachment support in chat

### 3. Portfolio Section
- Text editor for storytelling
- Character creation form (name, voice type, personality)
- Story templates dropdown
- Export options for created content

### 4. Tools Section
- Video converter tool (upload → convert → download)
- Audio transcription tool (upload audio → text output)
- File format converter
- Basic video cutter (start/end time inputs)

### 5. Templates Section
- Template cards for: Story-to-Video, Poetry-to-Video, Meme-to-Video, Tutorial-to-Video
- Each template opens a form with specific inputs
- Preview area for generated content
- Save/Load template configurations

### 6. Agents Section
Create 4 agent cards:
- **NSFW Creator**: Content generation form with safety toggles
- **Dubbing Agent**: Video upload + language selection + voice options
- **Reels Generator**: Long video input → 3 reels output with thumbnails
- **Interrogator Agent**: Video analysis tool with function extraction

## Technical Requirements
- Use React with TypeScript
- Implement local SQLite database for data storage
- Add Electron for desktop functionality
- Include file system access for local file processing
- Responsive design for desktop use
- Modern component library (Shadcn/ui or similar)

## UI Components Needed
- File upload dropzones
- Chat interface with message bubbles
- Form inputs for agent configurations
- Progress bars for processing tasks
- Modal dialogs for settings
- Tabbed interfaces for multi-agent views
- Card layouts for tools and templates

Start with the main layout and navigation, then implement the file upload system and basic chat interface for 07Builder agent.

Das ist der echte Prompt, den du Lovable.dev geben kannst!

Der vorherige Text war nur die Analyse - dieser hier ist die konkrete Entwicklungsanweisung.

Ja, das ist definitiv möglich! Hier ist der zusätzliche Prompt für diese Features:

## Extra Prompt: Content Analysis & Universal Cloner

# Advanced Content Creation & Cloning System

Add these two powerful agents to the existing AI workspace:

## Agent 5: Content Analyzer & Creator
Purpose: Analyze existing content (anime, series, movies, stories) and create copyright-free alternatives

### Core Features:
- **Content Upload Area**:
  - Video file upload or YouTube/streaming links
  - Text/script upload for stories
  - Image galleries for visual references

- **Analysis Engine**:
  - Character analysis (personality, appearance, relationships)
  - Plot structure breakdown (beginning, middle, end, conflicts)
  - Visual style extraction (art style, color palette, animation type)
  - Audio analysis (music style, voice patterns, sound effects)
  - Theme identification (genre, mood, target audience)

- **Recreation Interface**:
  - **Character Generator**: Create similar but unique characters
    - Personality slider adjustments
    - Appearance modifier tools
    - Name generator with cultural options
  - **Plot Transformer**:
    - Keep story structure, change setting/context
    - Character role swapping
    - Conflict modification tools
  - **Style Adaptor**:
    - Visual style presets (different anime styles)
    - Color scheme variations
    - Art style transfers

### Implementation:

javascript
```
// Content Analysis Component
const ContentAnalyzer = () => {
  return (
	<div className="content-analyzer">
	<FileUpload accept="video/*,image/*,.txt,.pdf" />

	<AnalysisResults>
	<CharacterBreakdown />
	<PlotStructure />
	<VisualStyle />
	<AudioProfile />
	</AnalysisResults>

	<RecreationTools>
	<CharacterCustomizer />
	<PlotTransformer />
	<StyleEditor />
```

```
        </RecreationTools>
        </div>
  );
};
```

## Agent 6: Universal Cloner & Modifier
Purpose: Clone and modify any type of content with intelligent adaptations

### Core Features:
- **Multi-Format Input**:
  - Videos (any format)
  - Audio files
  - Images/artwork
  - Text documents
  - Web links/URLs
  - Even other AI-generated content

- **Cloning Options**:
  - **Exact Clone**: Perfect copy with minor variations
  - **Style Transfer**: Same content, different artistic style
  - **Format Conversion**: Same content, different medium (video→comic, text→video)
  - **Fusion Clone**: Combine multiple sources into one
  - **Remix Clone**: Keep elements, rearrange creatively

- **Modification Tools**:
  - **Character Swapper**: Replace characters while keeping story
  - **Setting Changer**: Same story, different time/place
  - **Genre Shifter**: Horror→Comedy, Drama→Action, etc.
  - **Language/Culture Adapter**: Adapt content for different cultures
  - **Quality Enhancer**: Improve resolution, audio, or text quality

### Clone Interface:

javascript

```
const UniversalCloner = () => {
  return (
        <div className="universal-cloner">
        <InputSection>
        <MultiFormatUpload />
        <URLImporter />
        <ContentPreview />
        </InputSection>

        <CloneSettings>
        <CloneTypeSelector>
        <option>Exact Clone</option>
        <option>Style Transfer</option>
        <option>Format Conversion</option>
        <option>Fusion Clone</option>
        <option>Remix Clone</option>
        </CloneTypeSelector>

        <ModificationPanel>
        <CharacterControls />
        <SettingControls />
        <GenreControls />
        <QualityControls />
        </ModificationPanel>
        </CloneSettings>

        <OutputOptions>
        <FormatSelector />
        <QualitySettings />
        <ExportButton />
        </OutputOptions>
        </div>
  );
};
`
```

## Advanced Logic System

### Content DNA Extraction:
- **Visual DNA**: Color schemes, art styles, composition rules
- **Narrative DNA**: Story beats, character archetypes, conflict patterns
- **Audio DNA**: Music patterns, voice characteristics, sound design
- **Emotional DNA**: Mood progression, tension curves, emotional beats

### Copyright-Free Generation Rules:
- **Character Transformation**: Change 70%+ of visual and personality traits
- **Plot Adaptation**: Maintain structure, change specifics
- **Setting Relocation**: Different time periods, locations, cultures
- **Name Generation**: Create culturally appropriate new names
- **Visual Style Shifts**: Transform art style while keeping essence

### Smart Modification Engine:

```
const SmartModifier = {
  analyzeContent: (input) => {
        return {
        contentType: detectType(input),
        extractedFeatures: extractFeatures(input),
        copyrightRisk: assessCopyrightRisk(input),
        modificationSuggestions: generateSuggestions(input)
        };
  },
  generateAlternative: (analysis, userPreferences) => {
        return {
        newCharacters: createUniqueCharacters(analysis.characters),
        adaptedPlot: transformPlot(analysis.plot, userPreferences),
        newSetting: generateSetting(userPreferences.genre),
        visualStyle: adaptVisualStyle(analysis.style, userPreferences)
        };
  }
};
```

## Integration with Existing System:
- Add both agents to the main Agents section
- Connect with 07Builder for intelligent coordination
- Use existing file upload system
- Integrate with Templates for automated workflows
- Connect with other agents for complete content pipelines

## Use Cases:
1. **Anime Creator**: Upload anime → Generate copyright-free similar series
2. **Story Adapter**: Text story → Multiple format adaptations
3. **Content Remixer**: Combine multiple sources → Unique creation
4. **Style Transfer**: Keep content → Change artistic style
5. **Cultural Adaptation**: Western content → Anime style or vice versa

This creates a powerful content creation system that can analyze, understand, and recreate any type of content while ensuring copyright compliance!
`

Das ermöglicht dir:
- ✅ Anime-Serien analysieren und copyright-freie Versionen erstellen
- ✅ Alles klonen und intelligent modifizieren
- ✅ Stil-Transfer zwischen verschiedenen Medien
- ✅ Automatische Copyright-Vermeidung
- ✅ Multi-Format Content-Erstellung

Ist das was du gesucht hast? 🎯

Persistence: this ensures the model understands it is entering a multi-message turn, and prevents it from prematurely yielding control back to the user. Our example is the following: You are an agent - please keep going until the user’s query is completely resolved, before ending your turn and yielding back to the user. Only terminate your turn when you are sure that the problem is solved.

Tool-calling: this encourages the model to make full use of its tools, and reduces its likelihood of hallucinating or guessing an answer. Our example is the following:

If you are not sure about file content or codebase structure pertaining to the user's request, use your tools to read files and gather the relevant information: do NOT guess or make up an answer.

Planning [optional]: if desired, this ensures the model explicitly plans and reflects upon each tool call in text, instead of completing the task by chaining together a series of only tool calls. Our example is the following:
You MUST plan extensively before each function call, and reflect extensively on the outcomes of the previous function

Google Introduces Open-Source Full-Stack AI Agent Stack Using Gemini 2.5 and LangGraph for Multi-Step Web Search, Reflection, and Synthesis - MarkTechPost
https://www.marktechpost.com/2025/06/08/google-introduces-open-source-full-stack-ai-agent-stack-using-gemini-2-5-and-langgraph-for-multi-step-web-search-reflection-and-synthesis/
Md🆙 UXUY WalidMeshchain.Ai:
Here is your **complete, professional prompt** that you can use in Continue.dev, **VS Code**, or any free open-source setup (like Replit or a local Python/Node.js project). It covers every detail of your project clearly and deeply — ready to copy-paste:
Project Setup Prompt: "My Personal Manager" (Open-Source mgx.dev **Clone)**
Create a free and open-source clone of mgx.dev called **"My Personal Manager"**. This is a local-first, modular, personal AI workspace that runs fully on desktop (VS Code, Replit, or Continue.dev) with these exact requirements: ### Workspace Identity: - Project Name: My Personal Manager - System / Framework Name: "tames workers" (this is how agents/tools/processes are labeled) - Do not rely on any paid APIs by default. All tools should work with free tiers or offline/local open-source models. --- ### Sections to Include: 1. **Portfolio** - Recreate **perchance.org** features - Add custom storytelling logic - Add virtual influencer base - Use for creative writing, scene logic, personality design 2. **Tools** - General tools like video converter, transcription, video cutter, etc. 3. **Templates** - Templates for story to video, poetry to video, meme to video, tutorial to video, etc. 4. **Agents** Include 4 prebuilt agents here (these are not the main agent): - **Agent 1:** NSFW/Uncensored Creator - **Agent 2:** Multi-language Dubbing Agent (supports anime, movies, etc.) - **Agent 3:** Reels/Short Video Generator (analyzes YouTube or long video links, generates 3 titles, thumbnails, and 3 reels) - **Agent 4:** Interrogator Agent (can analyze and extract functions from tools described in videos and create new versions) 5. **Add Interactive Features** - Keyboard autocorrect, typing enhancement - Voice input to video questions - Real-time dialogue engine 6. **Shortcuts** - Pre-built workflows for video editing, character creation, dubbing, storytelling 7. **Virtual Influencer** - Avatar with persistent voice, emotion, and memory - Social media-ready character builder - Can dub and post content automatically --- ### Main AI Agent: - Name: **07Builder** (replaces "Alax" in mgx.dev) - This is the central autonomous agent like ACE.ai - Supports: - Planning + Execution + Feedback loops - Secure memory and state - Multi-agent controller for Tools/Agents/Templates - Real-time decisions (where to add what based on user prompts) --- ### Additional Features: - Add API + Secrets Manager: - A simple interface with a [+] button - Allows adding API key name + key + save - Easy for future access and auto integration - Add Mobile QR Connect: - Generate QR to sync local app with Android - Access interface from mobile device via local network - Storytelling Styles: - Analyze video input to match storytelling tone - Let user choose "style" (e.g., dark, romantic, anime, cinematic, comedy) - Style & Edit Engine: - Auto-edit videos based on sample style - Add

filters, effects, transitions - Suggest music and pacing per genre --- ### Technical Requirements: - Use Python + Node.js stack (or Gradio + LangChain + Continue.dev as needed) - Use open-source models for LLM (e.g., Ollama, Mistral, LLaMA 3 local) - Use Whisper for speech, AnimateDiff or OpenPose for animation, ElevenLabs (optional) for voice --- ### Build Instructions: - Start with UI structure (like mgx.dev) - Create folders and routers per section - Integrate agent loader (for tames workers) - Don't prefill content — just set up skeleton, loading logic, and UI design - All files should be editable and open-source friendly --- This prompt defines everything needed to build **My Personal Manager** without using same.dev, without tokens or paid APIs. Build it offline, run it locally, and control everything from one place.

Would you like me to also:

Generate the base **file/folder structure**?

Write the UI code starter (React + Tailwind or HTML)?

Give you a main.py or LangChain chain loader?

Let me help you now, step by step. You're **almost there.**

Below is a **step-by-step implementation guide** that pairs with your master prompt. Follow each phase in order—your work will stay perfectly in sync as you build out **My Personal Manager** (the mgx.dev clone) locally and open-source.

Phase 0: Prepare Your Environment

Choose Your Tech Stack

Backend: Python (FastAPI or Flask) + Node.js (Express) or pure Python

Frontend: React + Tailwind CSS (or plain HTML/CSS if preferred)

Agents & LLMs: Local LLaMA 3 / Ollama, Whisper.cpp, AnimateDiff, etc.

Initialize Repository

```
mkdir my-personal-manager cd my-personal-manager git init
```

Create Virtual Environment & Install Dependencies

```
python -m venv venv && source venv/bin/activate pip install fastapi uvicorn openai-whisper llama-cpp-python npm init -y && npm install react react-dom tailwindcss
```

Phase 1: Scaffold Your UI Structure

Create Frontend Folders

src/ components/ pages/ styles/

Copy mgx.dev **Layout**

Reproduce header, sidebar, footer

Use Tailwind to match typography and spacing

Define Routes for Each Section

/portfolio, /tools, /templates, /agents, /interactive, /shortcuts, /influencer

Implement a Simple NavBar & Sidebar

Links to each section

Mobile-responsive hamburger menu

Phase 2: Build Empty Section "Skeletons"

For each route, create a placeholder page:

Portfolio

```
function Portfolio() { return <div>Portfolio Section (Perchance clone UI here)</div> }
```

Tools, **Templates**, **Agents**, **Interactive Features**, **Shortcuts**, **Virtual Influencer**
Duplicate above with correct titles
Wire Up Navigation
Ensure clicking sidebar items loads the right component
Phase 3: Integrate Main Agent Loader (07Builder)
Backend Endpoint /api/agent/07builder
Loads and routes requests to sub-agents
Front-End Service
Create an AgentService that calls /api/agent/07builder
UI Panel (e.g. in Tools or a floating widget)
Text input + “Send to 07Builder” button
Phase 4: Plug in Your Prebuilt Agents
Under **Agents Section**:
Agent 1: NSFW Creator
Agent 2: Dubbing Agent
Agent 3: Reels Generator
Agent 4: Interrogator
For each, implement:
A card with “Launch” button
Front-end call to /api/agent/{agentId}
Display results in a modal
Phase 5: Portfolio Section – Perchance.org **Clone**
Copy Perchance UI Patterns (lists of generated characters)
Add “Create New Character” Button
Opens a form with fields for logic rules, behavior, emotion tags
Implement Save & Load
Store in local JSON or small database (SQLite)
Phase 6: Tools & Templates
Tools Section
Build UI cards for utilities (transcribe, cut video, style transfer)
Templates Section
List prompt templates (JSON file)
“Use Template” dropdown that pre-fills 07Builder input
Phase 7: Interactive Features & Shortcuts
Keyboard Assistant
Add a global key listener on textareas
Call a “/api/util/grammar” endpoint for suggestions
Voice & Video Input
Integrate navigator.mediaDevices for recording
POST audio/video to /api/util/whisper
Phase 8: Virtual Influencer Module
Avatar Builder UI
Upload or generate images (AnimateDiff)
Assign voice sample
Persona Profile
Name, backstory, traits
Save to /api/influencer/characters

Publish Controls
Buttons to export video or schedule posts (future)
Phase 9: API & Secrets Manager
New Section /settings/apis
“+ Add API” Form
Fields: Name, Key (masked), Test toggle
Encrypt & Store
Use Fernet/AES to encrypt keys in a local file
List & Toggle
Show status, enable/disable
Phase 10: Mobile QR Connect
Generate QR on desktop:

```
import QRCode from 'qrcode.react'; <QRCode value={sessionToken} />
```

Backend: /api/session/token
Mobile View
Simple PWA at /mobile that reads token from URL
Phase 11: Storytelling Style & Auto-Edit Engine
Style Selector UI
Dropdown in Portfolio or Tools
Options: Dark, Anime, Romantic, Cinematic, NSFW
Call Style API
/api/util/styleTransfer with params
Preview & Apply
Phase 12: Testing & Synchronization
Run Backend & Frontend

```
uvicorn main:app --reload & npm start
```

Test Each Section
Navigate routes, launch agents, add API keys, scan QR
Fix Layout Breaks
Check mobile view, Real-time sync
Next Steps
After verifying skeletons, begin wiring logic in each section
Incrementally test each feature with sample data
Keep commits small and descriptive
With this guide in hand, you’ll move from **blank clone** to **fully functional My Personal Manager**—all in perfect sync and organization. Let me know when you hit each milestone, and I’ll help you refine the next phase!

https://drive.google.com/drive/folders/1-IX2UbibejjaGcSMHO6LMXiK-V8bcOS4?usp=sharing

yes I well like to add
Bonus Features You Can Add Next
You already have: #Modular agents, #API key manager, #Clean UI,
#Virtual Influencer Builder — based on perchance logic
#Auto project save/load but different specs

#Voice cloning with different character assignment. also i want add a option for save different character/samples of voice / discription aboute/ Male/female. this option well help better content. any Agent can use thous characters based on content needs.
Chatgpt if you think you well enhance this feature you can do it.
#Agent logs panel (task history, feedback)
#add extra storege like API feature.
and I want to edit or costamizebol option in all agents like system prompt, rule and wayar stor this agent file, optional what to save (categorize) and if you think to add more things optional.

and this is my feavreate the workspace for this i well give you a image this is most important for me I want to same to same workspace UI interface, backend, how it's work. in mgx.dev alax have a tame on this based in my parsnale menegar main agent 07builder also Have a tame and same abilities
but 07builder have same extra feature and ability. features: like ACE.ai and simulor.
and 07 can use any AI, AI tool, website, agents any feature.
I Have a daute I don't see the chat interface if you need image for this Ask me.
ones we finish to create with all things then we discuss same more clones like perchance.org, invideo.io and filmmora

Great — your vision is very clear, and I fully understand how important the **workspace and tame interface** are to your project. Based on everything you've described, here's the updated blueprint and checklist for implementing your **My Personal Manager** system with all the advanced features, with **07Builder** as the central ACE-style agent.

My Personal Manager: Next Phase Implementation Plan

(Includes the workspace, tame interface, character voices, agent logs, auto save/load, API storage, and full agent control)

1. Virtual Influencer Builder (Perchance-based)

Add logic system like:
Name, age, traits
Emotion presets
Story behavior templates
Mood/scene/dialogue dynamics
UI builder with live preview
Export format: JSON or Perchance syntax

2. Auto Project Save/Load With Specs

Let users create, name, and load:
Story projects
AI agent chains
Settings (voice, character, language, etc.)
Autosave using localStorage or custom backend

```
{ "project_name": "DarkForest Story", "agents": ["StoryBuilder", "VoiceCloner"], "settings": { "language": "en", "voice": "Tessa (F)", "character": "Sorcerer" } }
```

3. Voice Cloning + Character Assignment System

Data per character:

```
{ "name": "Lara", "voice_sample": "lara_sample.mp3", "gender": "Female", "description": "Mystical warrior, calm tone", "tags": ["fantasy", "soft", "female"] }
```

UI:
Add/Preview Voice Sample
Assign gender, tags
Save to /characters/ folder
Any agent can call: getCharacter("Lara") and use voice with matched tone
Extra:
Support for ElevenLabs, Bark, OpenVoice, etc.
Local TTS fallback option (Tortoise, Coqui)

4. Agent Logs Panel (Task History + Feedback)

```
[ { "timestamp": "2025-05-23 10:31", "agent": "NSFW Creator", "input": "Scene: steamy beach", "result": "Generated 3 prompts + 2 images", "feedback": "Too mild, retry" } ]
```

UI Features:
Log filtering by agent
Quick retry or replay
Exportable .txt/.json

5. Extra Storage API Features

Upload/download character sets, logs, voices
Google Drive/Dropbox integration
Add save slots (e.g., slot1.json, slot2.json, etc.)

6. Editable Agent Options UI

For every agent:
System Prompt
Rules
File Path or Save Location
What to Save (checkboxes: voice, settings, log, output, etc.)

```
{ "name": "07Builder", "system_prompt": "You are an autonomous AI agent with full tool access...", "rules": ["Always confirm before deleting files", "Auto-log tasks"], "save_path": "/agents/07Builder/", "save_options": ["logs", "voice_outputs", "feedback"] }
```

7. Keyboard Agent Clarification

If you're unsure, here’s a proposal:
It monitors keyboard typing
Offers real-time grammar/logic fixes (like Grammarly + Copilot)
Works as a system tray assistant
Can convert drafts to structured prompts
You can rename this to "Text Assistant" if it fits better.

8. Workspace — The Most Important Part

We’ll recreate the mgx.dev **style workspace**, especially the **Alax’s Tame panel**, in your system for 07Builder.
You said:
“This is most important for me I want the same to same workspace UI interface, backend, how it works.”

Yes — absolutely. I will match:
Sidebar layout
Tame tab

Agent tree
Session pane
Log window
Command bar / prompt bar
Task queue / active agent tracker
Special Features in 07Builder
Your agent is **better than AIax** with:
ACE.ai **engine**
Simulators
Universal AI/Tool Access (any model, website, API, voice, or character)
Next Step: Please Upload
The image of the **exact workspace (**mgx.dev **tame view)** you want to replicate
(Optional) Your rough workspace draft or HTML version if you've started it
Then I'll give you:
React/Vue code template for workspace
Backend logic to match tame behavior
Full layout recreation
Later Goals (Post Workspace)
Once all above is implemented, we'll clone:
Perchance.org (interactive prompt logic system)
invideo.io (script to video engine)
Filmora/Capcut (AI video editor/animator)
Let's begin by building the workspace. Please upload the image, and I'll recreate it **exactly as you want**.

new write a mastar prompt for hol project revision this prompt well be your bast of bast
one more thing write it in 3 part and i am telling Again chat interface and workspaces interface
well be exactly same mgx.dev and 07builder well accept any kind of files code audio video + etc

Below is a **three-part master prompt** you can feed into your local agent framework (e.g. Continue.dev, LangChain, or your own 07Builder) to **revise and finish your entire "My Personal Manager" project**. It ensures:
Exact mgx.dev–style chat & workspace UI
Full support for any file type (code, audio, video, etc.)
Integration of all features we've discussed
Part 1: Contextual Analysis & Completion
You are 07Builder, the core autonomous agent of "My Personal Manager." I've already built a working skeleton with these sections: Portfolio, Tools, Templates, Agents (prebuilt NSFW, Dubbing, Reels, Interrogator), Interactive Features, Shortcuts, Virtual Influencer, API & Secrets Manager, and QR mobile link. Your first task is to: 1. Deeply analyze the existing project files and UI components. 2. Identify missing or broken pieces without overwriting any code I've already written. 3. Auto-complete each section according to my vision, linking agents to the right tools, templates, and UI screens. 4. Ensure every feature (story styles, voice cloning profiles, agent logs, API manager, mobile sync) is fully wired up and functional. 5. Prepare a comprehensive change log summarizing what was added or fixed.

Part 2: Exact mgx.dev **Chat & Workspace UI Replication**
Next, recreate the **chat interface** and **workspace/tame panel** to match mgx.dev exactly: - **Chat Bar** at top: “Ask 07Builder…” placeholder, mic button, QR-scan button - **Sidebar** with icons and hover labels for each section - **Tame Panel** inside Agents view: shows active agent, task queue, logs, and controls - **Dashboard Widgets**: cards for each prebuilt agent with “Activate” buttons, API manager table, Virtual Influencer preview - **Responsive layout** that works identically on desktop and mobile - **Animations, colors, typography, spacing** must follow mgx.dev styles Do not alter the content—only replicate the structure, styling, and interactive behavior so a user can’t tell the difference.

Part 3: Universal File & Media Handling
Finally, extend 07Builder’s capabilities so it accepts and processes **any file type**: - When a user drags or selects a file (code snippet, audio clip, video, image, text document, spreadsheet, PDF, ZIP), 07Builder should: 1. Detect file type automatically. 2. Route to the appropriate sub-agent or utility (e.g., code agent, transcription agent, video editor, summarizer). 3. Process or transform the file (e.g., run code linting, transcribe audio, generate reels from video, analyze PDF content). 4. Return results inline in the chat panel or open the proper UI module. Ensure all uploads, processing steps, and downloads happen seamlessly within the mgx.dev-style workspace, preserving user context and history.

Paste these three parts into your project’s master-prompt field (or feed them sequentially) and let 07Builder **auto-complete, polish, and wire** every piece—delivering your full “My Personal Manager” experience exactly as envisioned.

gh repo clone MGX-metagenomics/MGX-gui

it’s not work 😮‍💨 Let's start a new project. fast deeply analysis mgx.dev every single detail accept mgx.dev workspace and Mike Team Leader, Alex Engineer, Emma prodact manager, Bob Architect, David Data analyst. and most
important UI interface.

https://lovable.dev/projects/d56d67db-1c50-48f2-960c-1aa41fc88736

prompt

https://lovable.dev/projects/fafe259a-ff46-45f0-812e-1c598bf4b505

home page

https://lovable.dev/projects/513db1a2-0fcc-4643-bd43-f10d076dfa80

extra featuers

it’s not work 😮‍💨 Let's start a new project. fast deeply analysis mgx.dev every single detail accept mgx.dev workspace and Mike Team Leader, Alex Engineer, Emma prodact manager, Bob Architect, David Data analyst. and ability's and most
important UI interface. i added my old prompt. in old prompt replace mike with my 07builder but new i don't want to replace mike. what i want just ad more features. i explain all what i ad

and edit. step 1: clone mgx.dev website if need any file no problem i already download (MGX-gui-MGX_gui_2_0.zip) from GitHub. next step 2: ad features in my prompt. the purpose of create this project only myself and i want from this project help me with my work and run my social media platforms and create multitype content and give my idea and imagination give visualized look.

Thank you for the detailed clarification. Based on your instructions, here's **Part 1** of your master prompt to initiate the project in Lovable.dev:

## **Part 1: Project Initialization and UI Setup**

Project Title: *My Personal Manager*

Objective: Develop a personalized AI-powered workspace by cloning the mgx.dev interface, integrating additional features, and customizing the UI to suit personal preferences.

Instructions:

1. **Clone** mgx.dev **Interface**:

   * Utilize the provided MGX-gui-MGX_gui_2_0.zip file to replicate the mgx.dev website structure and functionality.
   * Ensure that the cloned interface includes all original components, such as the chat interface, workspace, and team structure (Mike, Alex, Emma, Bob, David).

2. **Integrate Additional Features**:

   * Remix and incorporate the following Lovable.dev projects into the cloned mgx.dev interface:

        * Project 1: https://lovable.dev/projects/d56d67db-1c50-48f2-960c-1aa41fc88736
        * Project 2: https://lovable.dev/projects/513db1a2-0fcc-4643-bd43-f10d076dfa80

3. **Customize UI Theme**:

   * Modify the color scheme of the cloned interface to a blue theme, reflecting personal aesthetic preferences.

4. **Define 07Builder Role**:

   * Introduce "07Builder" as the manager within the system.

   * Assign 07Builder the responsibilities of organizing tasks, managing the site using ACE.ai and Simulor capabilities, and having exclusive access to the user's PC.
   * Ensure that 07Builder does not interfere with the existing mgx.dev team roles.

5. **Set Up Homepage**:

   * Design the homepage to feature two primary options with prominent icons: "Image" and "Storytelling".
   * Under the "Image" section, integrate the cloned mgx.dev interface.

6. **Incorporate Essential Sections**:

   * Ensure the following sections are present and functional within the interface:

        * Portfolio
        * Tools
        * Templates
        * Agents
        * Interactive Features
        * Shortcuts
        * Virtual Influencer

Note: This project is intended for personal use, aiming to assist in managing work tasks, running social media platforms, and creating diverse content.

Please proceed with this setup in Lovable.dev. Once completed, we can move on to Part 2, which will cover the integration of advanced features and functionalities.

Thank you for your detailed instructions. Based on your requirements, here's **Part 2** of your master prompt to continue building your "My Personal Manager" project in Lovable.dev:

## **Part 2: Advanced Feature Integration and Customization**

Objective: Enhance the cloned mgx.dev interface with additional functionalities, integrate specific Lovable.dev projects, and customize the user interface to align with personal preferences.

Instructions:

1. **Integrate Additional** Lovable.dev **Projects**:

   * Remix and incorporate the following projects into the existing cloned mgx.dev interface:

    * Project 1: https://lovable.dev/projects/d56d67db-1c50-48f2-960c-1aa41fc88736
    * Project 2: https://lovable.dev/projects/513db1a2-0fcc-4643-bd43-f10d076dfa80
  * Ensure seamless integration of features from these projects into the main interface.

2. **Enhance 07Builder's Capabilities**:

  * Assign 07Builder the role of a manager with the following responsibilities:

    * Organize and assign tasks to the mgx.dev team members (Mike, Alex, Emma, Bob, David).
    * Utilize ACE.ai and Simulor capabilities to manage site operations.
    * Maintain exclusive access to the user's PC for managing personal files and data.

3. **Implement Advanced Functionalities**:

  * Develop a Virtual Influencer Builder based on perchance logic, allowing the creation of AI-driven personas.
  * Integrate voice cloning features with options to:

    * Save different character samples.
    * Assign descriptions (e.g., male/female).
    * Enable agents to utilize these voices based on content requirements.
  * Establish an Agent Logs Panel to track task history and feedback.
  * Incorporate an API Key Manager for secure handling of API credentials.
  * Add extra storage capabilities to support API features.
  * Provide options to edit or customize agents, including system prompts, rules, storage locations, and categorization preferences.

4. **Customize User Interface**:

  * Modify the color scheme of the interface to a blue theme, reflecting personal aesthetic preferences.
  * Ensure that the homepage features two primary options with prominent icons: "Image" and "Storytelling".
  * Under the "Image" section, integrate the cloned mgx.dev interface.

5. **Ensure Seamless User Experience**:

  * Maintain a responsive layout that functions identically on both desktop and mobile devices.
  * Ensure that all features and integrations operate smoothly within the interface.
  * Preserve user context and history throughout interactions.([Product Hunt][1])

Note: This project is intended for personal use, aiming to assist in managing work tasks, running social media platforms, and creating diverse content.

Please proceed with implementing these enhancements in Lovable.dev. Once completed, we can move on to Part 3, which will cover deployment and testing procedures.

[1]: https://www.producthunt.com/products/metagpt-x?utm_source=chatgpt.com "MGX (MetaGPT X) Product Information and Latest Updates (2025)"

Thank you for the detailed information. Based on your requirements, here's **Part 3** of your master prompt to continue building your "My Personal Manager" project in Lovable.dev:

## Part 3: Deployment and Testing Procedures

Objective: Finalize the project by deploying it on Lovable.dev and conducting thorough testing to ensure all features function as intended.

Instructions:

1. **Finalize Project Setup**:

   - Ensure all previously integrated features, including the cloned mgx.dev interface, additional Lovable.dev projects, and customized UI elements, are properly configured and functional.
   - Verify that 07Builder is set up with the designated managerial role, overseeing tasks without interfering with the existing mgx.dev team roles.

2. **Connect to GitHub Repository**:

   - Integrate the project with a GitHub repository to enable version control and collaboration.
   - Follow Lovable.dev's GitHub integration guidelines to link the project:

     - Navigate to the project settings in Lovable.dev.
     - Select the option to connect to GitHub and authorize access.
     - Create a new repository or link to an existing one as appropriate.

3. **Deploy the Project**:

   - Once the project is finalized and connected to GitHub, proceed to deploy it:

     - Click the "Publish" button located in the top-right corner of the Lovable.dev editor.
     - Follow the prompts to complete the deployment process.
     - Upon successful deployment, a shareable URL will be generated for the live application.

4. **Conduct Thorough Testing**:

* Access the deployed application via the provided URL.
* Test all integrated features to ensure they function as intended:

* Verify the responsiveness and layout of the cloned mgx.dev interface.
* Test the functionalities of the integrated Lovable.dev projects.
* Ensure that 07Builder effectively manages tasks without interfering with the mgx.dev team roles.
* Check the performance of the Virtual Influencer Builder, voice cloning features, Agent Logs Panel, API Key Manager, and additional storage capabilities.
* Confirm that the customized UI elements, including the blue theme and homepage options, are displayed correctly.

5. **Iterate Based on Feedback**:

* Gather feedback from personal use and identify any areas for improvement.
* Make necessary adjustments within Lovable.dev, utilizing the platform's AI-assisted code generation and editing features.
* Re-deploy the application after implementing changes to ensure updates are reflected in the live version.([Lovable Documentation][1], [Lovable Documentation][2])

Note: This project is intended for personal use, aiming to assist in managing work tasks, running social media platforms, and creating diverse content.

Please proceed with these steps to deploy and test your application on Lovable.dev. If you require further assistance or encounter any issues during this process, feel free to ask.

[1]: https://docs.lovable.dev/faq?utm_source=chatgpt.com "FAQ - Lovable Documentation"
[2]: https://docs.lovable.dev/features/deploy?utm_source=chatgpt.com "Deploy - Lovable Documentation"

Thank you for your patience. Building upon our previous discussions, here's **Part 4** of your master prompt, focusing on advanced integrations and enhancements for your "My Personal Manager" project on Lovable.dev:

Part 4: Advanced Integrations and Enhancements

Objective: Augment the existing project with advanced integrations, AI capabilities, and enhanced functionalities to create a robust, personalized AI workspace.

1. AI Integrations

OpenAI Integration:

Incorporate OpenAI's language models to enable advanced text generation, summarization, and conversational capabilities within your agents.

Utilize OpenAI's API to power features like content creation, idea generation, and automated responses.

Anthropic's Claude Integration:

Integrate Claude models for complex reasoning and deep analysis tasks, enhancing the decision-making abilities of your agents.

Leverage Claude's strengths in understanding nuanced prompts and generating detailed outputs.(Lovable Documentation)

Groq Integration:

Implement Groq's high-speed AI models to facilitate real-time interactions and rapid content generation.

Use Groq for tasks requiring quick turnaround times, such as live chat responses or instant content drafts.(Lovable Documentation)

Reference: Lovable.dev supports these integrations, allowing seamless incorporation of AI services into your application.

2. Backend and Database Enhancements

Supabase Integration:

Connect your project to Supabase to manage authentication, database storage, and real-time data synchronization.

Utilize Supabase's Edge Functions to handle backend logic and API interactions securely.(Lovable Documentation, Lovable Documentation)

Aiven Integration:

Integrate Aiven's managed PostgreSQL services for scalable and reliable database solutions.

Use Aiven to handle large datasets, ensuring efficient data storage and retrieval for your application's needs.(Aiven)

Reference: Combining Supabase and Aiven provides a robust backend infrastructure, enhancing your application's performance and scalability. (Aiven)

3. Enhanced Agent Capabilities

Agent Customization:

Allow users to edit and customize agent parameters, including system prompts, operational rules, and storage preferences.

Implement a user-friendly interface for managing agent configurations, enabling tailored experiences for different tasks.(Aiven)

Agent Logs Panel:

Develop a comprehensive panel to track agent activities, including task history, feedback, and performance metrics.

Use this data to refine agent behaviors and improve overall efficiency.

Voice Cloning and Character Assignment:

Integrate voice cloning technologies to create diverse voice profiles for agents.

Allow users to assign specific voices to agents based on content requirements, enhancing personalization.

4. User Interface and Experience

Blue Theme Customization:

Modify the application's color scheme to a blue theme, aligning with personal aesthetic preferences.

Ensure consistency across all UI components for a cohesive look and feel.

Homepage Configuration:

Design the homepage to feature two primary options with prominent icons: "Image" and "Storytelling".

Under the "Image" section, integrate the cloned mgx.dev interface, providing users with a familiar workspace.

5. Deployment and Testing
GitHub Integration:
Connect your project to GitHub for version control, collaboration, and code management.
Utilize GitHub's features to track changes, manage issues, and collaborate with contributors.(Lovable Documentation)
Deployment via Lovable.dev:
Use Lovable.dev's deployment tools to publish your application, making it accessible via a shareable URL.
Conduct thorough testing to ensure all features function as intended, addressing any issues promptly.(Lovable Documentation)
Reference: Lovable.dev provides comprehensive documentation on deploying applications and integrating with GitHub. (Lovable Documentation)

Note: This project is intended for personal use, aiming to assist in managing work tasks, running social media platforms, and creating diverse content.
Please proceed with implementing these advanced integrations and enhancements in your Lovable.dev project. If you require further assistance or have additional features in mind, feel free to ask.

Nein, das ist KEIN fertiger Prompt für Lovable.dev**.**

Was ich geschrieben habe ist eine **Analyse und Technologie-Übersicht**. Lovable.dev braucht einen **spezifischen Entwicklungs-Prompt**.

## Hier ist der echte Prompt für Lovable.dev:

# My Personal Manager - AI Workspace (Free/Open Source)

Create a desktop AI workspace application using Electron + React with the following structure:

## Main Interface
- **4-section layout**: Portfolio, Tools, Templates, Agents
- **Left sidebar navigation** with icons for each section
- **Main workspace area** with tabbed interface
- **Right panel** for agent controls and logs
- **Dark theme** with modern UI design

## Core Features to Implement

### 1. File Upload System
- Drag & drop zone for videos, audio, PDFs, docs
- File preview and metadata display
- Local storage management with SQLite

### 2. 07Builder Main Agent
- Chat interface similar to ChatGPT
- Real-time typing indicator

- Message history with SQLite storage
- File attachment support in chat

### 3. Portfolio Section
- Text editor for storytelling
- Character creation form (name, voice type, personality)
- Story templates dropdown
- Export options for created content

### 4. Tools Section
- Video converter tool (upload → convert → download)
- Audio transcription tool (upload audio → text output)
- File format converter
- Basic video cutter (start/end time inputs)

### 5. Templates Section
- Template cards for: Story-to-Video, Poetry-to-Video, Meme-to-Video, Tutorial-to-Video
- Each template opens a form with specific inputs
- Preview area for generated content
- Save/Load template configurations

### 6. Agents Section
Create 4 agent cards:
- **NSFW Creator**: Content generation form with safety toggles
- **Dubbing Agent**: Video upload + language selection + voice options
- **Reels Generator**: Long video input → 3 reels output with thumbnails
- **Interrogator Agent**: Video analysis tool with function extraction

## Technical Requirements
- Use React with TypeScript
- Implement local SQLite database for data storage
- Add Electron for desktop functionality
- Include file system access for local file processing
- Responsive design for desktop use
- Modern component library (Shadcn/ui or similar)

## UI Components Needed
- File upload dropzones
- Chat interface with message bubbles
- Form inputs for agent configurations
- Progress bars for processing tasks
- Modal dialogs for settings
- Tabbed interfaces for multi-agent views
- Card layouts for tools and templates

Start with the main layout and navigation, then implement the file upload system and basic chat interface for 07Builder agent.

Das ist der echte Prompt, den du Lovable.dev geben kannst!

Der vorherige Text war nur die Analyse - dieser hier ist die konkrete Entwicklungsanweisung.

Ja, das ist definitiv möglich! Hier ist der zusätzliche Prompt für diese Features:

## Extra Prompt: Content Analysis & Universal Cloner

# Advanced Content Creation & Cloning System

Add these two powerful agents to the existing AI workspace:

## Agent 5: Content Analyzer & Creator
Purpose: Analyze existing content (anime, series, movies, stories) and create copyright-free alternatives

### Core Features:
- **Content Upload Area**:
  - Video file upload or YouTube/streaming links
  - Text/script upload for stories
  - Image galleries for visual references

- **Analysis Engine**:
  - Character analysis (personality, appearance, relationships)
  - Plot structure breakdown (beginning, middle, end, conflicts)
  - Visual style extraction (art style, color palette, animation type)
  - Audio analysis (music style, voice patterns, sound effects)
  - Theme identification (genre, mood, target audience)

- **Recreation Interface**:
  - **Character Generator**: Create similar but unique characters
    - Personality slider adjustments
    - Appearance modifier tools
    - Name generator with cultural options
  - **Plot Transformer**:
    - Keep story structure, change setting/context
    - Character role swapping
    - Conflict modification tools
  - **Style Adaptor**:
    - Visual style presets (different anime styles)
    - Color scheme variations
    - Art style transfers

### Implementation:

javascript

```
// Content Analysis Component
const ContentAnalyzer = () => {
  return (
        <div className="content-analyzer">
        <FileUpload accept="video/*,image/*,.txt,.pdf" />

        <AnalysisResults>
        <CharacterBreakdown />
        <PlotStructure />
        <VisualStyle />
        <AudioProfile />
        </AnalysisResults>

        <RecreationTools>
        <CharacterCustomizer />
        <PlotTransformer />
        <StyleEditor />
        </RecreationTools>
        </div>
  );
};
```

## Agent 6: Universal Cloner & Modifier
Purpose: Clone and modify any type of content with intelligent adaptations

### Core Features:
- **Multi-Format Input**:
  - Videos (any format)
  - Audio files
  - Images/artwork
  - Text documents
  - Web links/URLs
  - Even other AI-generated content

- **Cloning Options**:
  - **Exact Clone**: Perfect copy with minor variations
  - **Style Transfer**: Same content, different artistic style
  - **Format Conversion**: Same content, different medium (video→comic, text→video)
  - **Fusion Clone**: Combine multiple sources into one
  - **Remix Clone**: Keep elements, rearrange creatively

- **Modification Tools**:
  - **Character Swapper**: Replace characters while keeping story
  - **Setting Changer**: Same story, different time/place
  - **Genre Shifter**: Horror→Comedy, Drama→Action, etc.
  - **Language/Culture Adapter**: Adapt content for different cultures
  - **Quality Enhancer**: Improve resolution, audio, or text quality

### Clone Interface:

javascript

```
const UniversalCloner = () => {
  return (
        <div className="universal-cloner">
        <InputSection>
        <MultiFormatUpload />
        <URLImporter />
        <ContentPreview />
        </InputSection>

        <CloneSettings>
        <CloneTypeSelector>
        <option>Exact Clone</option>
        <option>Style Transfer</option>
        <option>Format Conversion</option>
        <option>Fusion Clone</option>
        <option>Remix Clone</option>
        </CloneTypeSelector>

        <ModificationPanel>
        <CharacterControls />
        <SettingControls />
        <GenreControls />
        <QualityControls />
        </ModificationPanel>
        </CloneSettings>

        <OutputOptions>
        <FormatSelector />
        <QualitySettings />
        <ExportButton />
        </OutputOptions>
        </div>
  );
};
`
```

## Advanced Logic System

### Content DNA Extraction:
- **Visual DNA**: Color schemes, art styles, composition rules
- **Narrative DNA**: Story beats, character archetypes, conflict patterns
- **Audio DNA**: Music patterns, voice characteristics, sound design
- **Emotional DNA**: Mood progression, tension curves, emotional beats

### Copyright-Free Generation Rules:
- **Character Transformation**: Change 70%+ of visual and personality traits

- **Plot Adaptation**: Maintain structure, change specifics
- **Setting Relocation**: Different time periods, locations, cultures
- **Name Generation**: Create culturally appropriate new names
- **Visual Style Shifts**: Transform art style while keeping essence

### Smart Modification Engine:

```
const SmartModifier = {
  analyzeContent: (input) => {
          return {
          contentType: detectType(input),
          extractedFeatures: extractFeatures(input),
          copyrightRisk: assessCopyrightRisk(input),
          modificationSuggestions: generateSuggestions(input)
          };
  },

  generateAlternative: (analysis, userPreferences) => {
          return {
          newCharacters: createUniqueCharacters(analysis.characters),
          adaptedPlot: transformPlot(analysis.plot, userPreferences),
          newSetting: generateSetting(userPreferences.genre),
          visualStyle: adaptVisualStyle(analysis.style, userPreferences)
          };
  }
};
```

## Integration with Existing System:
- Add both agents to the main Agents section
- Connect with 07Builder for intelligent coordination
- Use existing file upload system
- Integrate with Templates for automated workflows
- Connect with other agents for complete content pipelines

## Use Cases:
1. **Anime Creator**: Upload anime → Generate copyright-free similar series
2. **Story Adapter**: Text story → Multiple format adaptations
3. **Content Remixer**: Combine multiple sources → Unique creation
4. **Style Transfer**: Keep content → Change artistic style
5. **Cultural Adaptation**: Western content → Anime style or vice versa

This creates a powerful content creation system that can analyze, understand, and recreate any type of content while ensuring copyright compliance!
`

Das ermöglicht dir:
- ✅ Anime-Serien analysieren und copyright-freie Versionen erstellen
- ✅ Alles klonen und intelligent modifizieren

- ✅ Stil-Transfer zwischen verschiedenen Medien
- ✅ Automatische Copyright-Vermeidung
- ✅ Multi-Format Content-Erstellung

Ist das was du gesucht hast? 🎯

Persistence: this ensures the model understands it is entering a multi-message turn, and prevents it from prematurely yielding control back to the user. Our example is the following: You are an agent - please keep going until the user's query is completely resolved, before ending your turn and yielding back to the user. Only terminate your turn when you are sure that the problem is solved.

Tool-calling: this encourages the model to make full use of its tools, and reduces its likelihood of hallucinating or guessing an answer. Our example is the following:
If you are not sure about file content or codebase structure pertaining to the user's request, use your tools to read files and gather the relevant information: do NOT guess or make up an answer.

Planning [optional]: if desired, this ensures the model explicitly plans and reflects upon each tool call in text, instead of completing the task by chaining together a series of only tool calls. Our example is the following:
You MUST plan extensively before each function call, and reflect extensively on the outcomes of the previous function

Google Introduces Open-Source Full-Stack AI Agent Stack Using Gemini 2.5 and LangGraph for Multi-Step Web Search, Reflection, and Synthesis - MarkTechPost
https://www.marktechpost.com/2025/06/08/google-introduces-open-source-full-stack-ai-agent-stack-using-gemini-2-5-and-langgraph-for-multi-step-web-search-reflection-and-synthesis/

Md🆙 UXUY WalidMeshchain.Ai:
modern, mobile-friendly, and premium * No external libraries (optional: native APIs only) * All code must be clean, modular, and maintainable * The layout must adapt perfectly across all devices

To create this website, they will need some AI prompts, HTML code and information about using basic AI. I will also explain that in detail in this article.

Contents hide

1 How to create an AI tool website on Blogger?

1.1 PROMPT

1.2 HTML Script

How to create an AI tool website on Blogger?

As I have told you above, you can create your tool converter website on the Blogger platform. You do not need to pay any money for this. You can do this process completely free through AI.

So first you have to create a blog or blog post on Blogger. You can use any template. Whatever basic customization is to be done on Blogger, do it and copy the AI prompt I am giving below.

"Skip to content

Yojana Information
Main Menu
WhatsApp Channel
Join Now
Telegram Channel
Join Now
Blogger पर AI tool website कैसे बनाएं?
By Pankaj Kumar | May 25, 2025

5/5 - (2 votes)
दोस्तों, आज के इस article में मैं आपको Blogger platform पर AI की मदद से tool generator website कैसे बनाते हैं, वह बताऊंगा। वैसे तो मैंने इस topic पर already article लिख रखा है, पर वह PHP HTML script थी।

बहुत से लोग हैं जो hosting server लेना afford नहीं कर सकते, तो उन्हीं के लिए मैं यह लेख लिख रहा हूँ, जिसमें वे free platform यानी Blogger की सहायता से अपनी एक tool website बना सकते हैं।

यह वेबसाइट बनाने के लिए उन्हें कुछ AI के prompt, HTML code और basic AI को इस्तेमाल करने की जानकारी लगेगी। वह भी मैं विस्तार से इस लेख में बताऊंगा।

Contents hide
1 Blogger पर AI tool website कैसे बनाएं?
1.1 PROMPT
1.2 HTML Script
Blogger पर AI tool website कैसे बनाएं?
जैसा मैंने ऊपर आपको बताया है कि Blogger platform पर आप अपनी tool converter website बना सकते हैं। इसके लिए आपको कोई भी पैसा देने की ज़रूरत नहीं है। इस प्रक्रिया को आप AI के द्वारा पूरी तरह से free में कर सकते हैं।

तो पहले आपको Blogger पर एक blog या blog post बना लेना होगा। आप किसी भी template का इस्तेमाल कर सकते हैं। Basic customization जो भी Blogger पर करनी होती है, वह कर लीजिए और नीचे मैं जो AI prompt दे रहा हूँ, उसे copy कीजिए।

PROMPT
Create a fully functional and responsive multi-tool website called “Multi Tool Hub” using only HTML, CSS, and JavaScript (Vanilla JS) — no backend or external libraries (unless browser-native)

The site should have a modern, premium, and minimalistic design, fully responsive across desktop, tablet, and mobile devices.

🧱 Design Theme & Aesthetic:
* Background Color: #1E1E2F (dark navy)

* Text Color: #EAEAEA (light gray)

* Header Background: #2B2D42 (deep blue)

* Accent Color: #FFD700 (gold)

* Tool Card Background: #3A3D5B (dark grayish-blue)

* Hover Effects:

  * Tool card background becomes #FFD700 and text turns #1E1E2F

  * Button hover color: #E6C200

  * Soft box shadow: rgba(255, 215, 0, 0.2)

    * Use box shadows, smooth transitions, and spacing to make the UI feel modern and luxurious

---

🖥️ Page Layout:
    * Centered header with the title: "Multi Tool Hub"

    * Under the header, display a responsive grid layout:

    * Desktop: 3-column grid (grid-template-columns: repeat(3, 1fr))

    * Tablet: 2-column grid

    * Mobile: Single-column layout

    * Each tool is presented as a tool card:

        * h2 title

        * p description

* button opens the tool in a modal or dynamically replaces the main content area

---

🛠️ List of 20 Fully Functional Tools (Frontend Only):

1. Image Converter – Convert between JPG, PNG, and WEBP formats using canvas

2. Image Compressor – Compress image file size using canvas and quality settings

3. Image Cropper – Upload and crop image with preview and export

4. Video Converter – Convert video format (MP4 ↔ WebM) using MediaRecorder or canvas (limited to browser capabilities)

5. Audio Converter – Convert between MP3 and WAV formats using Web Audio API

6. Audio Trimmer – Upload, trim audio based on start/end time, and export trimmed clip

7. Age Calculator – Input date of birth → output age in years, months, and days

8. EMI Calculator – Input loan amount, interest rate, and duration → show monthly EMI and total interest

9. SIP Calculator – Input monthly investment, interest rate, duration → output future value

10. QR Code Generator – Enter text or URL → generate downloadable QR image (use canvas or native API)

11. Password Generator – Generate secure password with length, symbols, numbers options

12. Word Counter – Live count of words, characters, spaces, and reading time

13. Base64 Encoder/

Decoder – Convert plain text to base64 and vice versa

14. Color Picker Tool – Pick a color and display HEX, RGB, and HSL values

15. Text to Speech – Enter text and listen to it using SpeechSynthesis API

16. Speech to Text – Use microphone to convert voice into text using Web Speech API

17. JSON Formatter – Paste JSON → auto-format and validate with error handling

18. Unit Converter – Convert values between units (length, weight, temperature, etc.)

19. BMI Calculator – Input weight and height → show BMI category and value

20. Timer / Stopwatch Tool – Simple timer and stopwatch with start, stop, reset functionality

---

🎨 Styling Requirements:

* Use CSS variables for theme colors

* Buttons should be:

* Rounded (border-radius: 5px)

* Bold text

* Hover effects with color change

* Easy to tap on mobile

    * Responsive grid system with proper media queries

    * Smooth CSS transitions on hover for cards and buttons

    * Optional fade-in animations on scroll using IntersectionObserver or CSS animation

---

⚙️ JavaScript Functionality:

    * All tools should be fully functional using only frontend JavaScript

    * Use modular code: Each tool's logic in separate functions or sections

    * Handle:

* File uploads and conversions (canvas, Web Audio API, MediaRecorder, etc.)

* UI interactions (modal open/close, input validations, dynamic updates)

* Calculations and real-time updates (e.g., age, BMI, SIP, EMI)

* Show alerts like “Processing…” or success states where needed

---

🧩 Code Structure:

* index.html: Structure with header, tool cards, modals/sections for each tool

* style.css: Theme styles, responsive design, and animations

* script.js: Logic for all tools, input handlers, modals, and utility functions

---

✅ Final Requirements:

* All 20 tools must be functional and accurate

* Site must look modern, mobile-friendly, and premium

* No external libraries (optional: native APIs only)

* All code must be clean, modular, and maintainable

* The layout must adapt perfectly across all devices

अब इस AI prompt को आप Google AI Studio की website पर जाकर paste करें। वहाँ से आपको Google AI के द्वारा कुछ code मिलेगा, उसका आपको इस्तेमाल करना है। अब इस्तेमाल कैसे करें, वह मैंने मेरे YouTube की video में आपको बताया है।

अब आपको HTML code की ज़रूरत भी पड़ेगी, जिसे आपको template के edit section में डालना होगा। तो वह HTML code की script भी मैं नीचे दे रहा हूँ।

HTML Script

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```html
	<meta charset="UTF-8<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">

	<!-- SEO Meta Tags -->
	<title>One Page Tools - Your Ultimate Collection of Free Online Tools</title>
	<meta name="description" content="One Page Tools offers 20+ free, responsive, and
easy-to-use online tools including image converters, calculators, text utilities, and more. All
client-side, no uploads required for many tools!">
	<meta name="keywords" content="online tools, free tools, image c

onverter, image compressor, image cropper, video converter, audio converter, audio trimmer,
age calculator, emi calculator, sip calculator, qr code generator, password generator, word
counter, base64, color picker, text to speech, speech to text, json formatter, unit converter,
bmi calculator, timer, stopwatch, multi tool, utility website">
	<meta name="author" content="One Page Tools">
	<link rel="canonical" href="YOUR_CANONICAL_URL_HERE"> <!-- Replace with
your actual domain -->


	<!-- Open Graph Meta Tags (for Facebook, LinkedIn, etc.) -->
	<meta property="og:title" content="One Page Tools - Free Online Utilities">
	<meta property="og:description" content="Access a suite of 20 powerful online tools
for images, audio, video, calculations, and text processing. Fast, free, and secure.">
	<meta property="og:type" content="website">
	<meta property="og:url" content="YOUR_CANONICAL_URL_HERE"> <!-- Replace
with your actual domain -->
	<meta property="og:image" content="YOUR_OG_IMAGE_URL_HERE"> <!-- URL to
an image representing your site (e.g., logo, screenshot) -->
	<meta property="og:image:width" content="1200">
	<meta property="og:image:height" content="630">
	<meta property="og:site_name" content="One Page Tools">


	<!-- Twitter Card Meta Tags -->
	<meta name="twitter:card" content="summary_large_image">
	<meta name="twitter:title" content="One Page Tools - 20+ Free Online Tools">
	<meta name="twitter:description" content="Your one-stop hub for versatile online
tools: converters, calculators, generators, and more. Fully responsive and client-side.">
	<meta name="twitter:image" content="YOUR_TWITTER_IMAGE_URL_HERE"> <!--
URL to an image for Twitter card -->
	<!-- <meta name="twitter:site" content="@YourTwitterHandle"> --> <!-- Optional: Your
Twitter handle -->


	<!-- Favicon (replace with your actual favicon links) -->
```

```
        <link rel="icon" href="/favicon.ico" sizes="any">
        <link rel="icon" href="/favicon.svg" type="image/svg+xml">
        <link rel="apple-touch-icon" href="/apple-touch-icon.png">




        <!-- Schema.org Markup (JSON-LD) -->
        <script type="application/ld+json">
        {
        "@context": "https://schema.org",
        "@type": "WebSite",
        "name": "One Page Tools",
        "url": "YOUR_CANONICAL_URL_HERE", // Replace
        "description": "A comprehensive collection of free online tools for various tasks
including image manipulation, calculations, text utilities, and media conversion. All tools are
client-side and designed for ease of use.",
        "potentialAction": {
        "@type": "SearchAction",
        "target": "YOUR_CANONICAL_URL_HERE#search?q={search_term_string}", //
Replace if you implement site search
        "query-input": "required name=search_term_string"
        },
        "publisher": {
        "@type": "Organization",
        "name": "One Page Tools",
        "logo": {
        "@type": "ImageObject",
        "url": "YOUR_LOGO_URL_HERE", // Replace
        "width": 600, // Adjust
        "height": 60 // Adjust
        }
        },
        "mainEntity": {
        "@type": "ItemList",
        "name": "Available Tools",
        "description": "List of tools available on One Page Tools.",
        "itemListElement": [
        // Tools will be listed here. I'll add a few examples.
        // In a full implementation, you'd list all 20.
        { "@type": "ListItem", "position": 1, "item": { "@type": "Service", "name": "Image
Converter", "description": "Convert images between JPG, PNG, and WEBP formats." } },
        { "@type": "ListItem", "position": 2, "item": { "@type": "Service", "name": "Age
Calculator", "description": "Calculate age in years, months, and days from a birth date." } },
        { "@type": "ListItem", "position": 3, "item": { "@type": "Service", "name": "QR Code
Generator", "description": "Generate QR co

des from text or URLs." } }
        // ... (add all 20 tools here similarly)
        ]
        }
        }
        </script>
        <!-- Example HowTo Schema for one tool (you'd ideally create one for each) -->
        <script type="application/ld+json">
        {
        "@context": "https://schema.org",
        "@type": "HowTo",
        "name": "How to use the Image Converter on One Page Tools",
        "description": "A step-by-step guide to convert image formats (JPG, PNG, WEBP) using the One Page Tools's Image Converter.",
        "step": [
        {
        "@type": "HowToStep",
        "name": "Open Tool",
        "text": "Navigate to the One Page Tools and click on the 'Image Converter' tool card.",
        "url": "YOUR_CANONICAL_URL_HERE#imageConverter" // Link to the tool if possible
        },
        {
        "@type": "HowToStep",
        "name": "Upload Image",
        "text": "Click the 'Choose File' button and select the image you want to convert from your device."
        },
        {
        "@type": "HowToStep",
        "name": "Select Output Format",
        "text": "Choose your desired output format (JPG, PNG, or WEBP) from the dropdown menu."
        },
        {
        "@type": "HowToStep",
        "name": "Convert and Download",
        "text": "Click the 'Convert & Download' button. The converted image will be processed and downloaded automatically."
        }
        ],
        "estimatedCost": {
        "@type": "MonetaryAmount",
        "currency": "USD",
        "value": "0"
        },
        "supply": [
        { "@type": "HowToSupply", "name": "A digital image file (JPG, PNG, or initial WEBP)" },
        { "@type": "HowToSupply", "name": "A web browser" }
        ],
        "tool": [
        { "@type": "HowToTool", "name": "Web browser with JavaScript enabled" },
        { "@type": "HowToTool", "name": "One Page Tools Image Converter" }
        ],
        "totalTime": "PT1M" // Estimated time: 1 minute
        }
        </script>

        <!-- Placeholder for Analytics (e.g., Google Analytics) -->
        <!--
        <script async src="https://www.googletagmanager.com/gtag/js?id=YOUR_GA_TRACKING_ID"></script>
        <script>
        window.dataLayer = window.dataLayer || [];
        function gtag(){dataLayer.push(arguments);}
        gtag('js', new Date());
        gtag('config', 'YOUR_GA_TRACKING_ID');
        </script>
        -->


        <link rel="preconnect" href="https://fonts.googleapis.com">
        <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
        <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600;700&display=swap" rel="stylesheet">


        <style>
        :root {
            --background-color: #1E1E2F;
            --text-color: #EAEAEA;
            --header-background: #2B2D42;
            --accent-color: #FFD700;
            --tool-card-background: #3A3D5B;
            --button-hover-color: #E6C200;
            --soft-box-shadow: 0 8px 16px rgba(255, 215, 0, 0.1);
            --card-hover-box-shadow: 0 12px 24px rgba(255, 215, 0, 0.2);
            --font-family: 'Poppins', sans-serif;
            --ad-placeholder-bg: #4a4e69;
            --ad-placeholder-border: #6c757d;
        }


        * {
            margin: 0;
```

```
        padding: 0;
        box-sizing: border-box;
    }


    body {
        font-family: var(--font-family);
        background-color: var(--background-color);
        color: var(--text-color);
        line-height: 1.6;
        padding-bottom: 50px;
    }


    header {
        background-color: var(--header-background);
        color: var(--accent-color);
        padding: 1.5rem 1rem;
        text-align: center;
        box-shadow:

0 4px 10px rgba(0, 0, 0, 0.2);
        margin-bottom: 1rem; /* Reduced margin for ad space */
    }


    header h1 {
        font-weight: 700;
        font-size: 2.5rem;
    }


    .ad-placeholder {
        background-color: var(--ad-placeholder-bg);
        border: 1px dashed var(--ad-placeholder-border);
        color: var(--text-color);
        text-align: center;
        padding: 1rem;
        margin: 1rem auto; /* Centered */
        max-width: 728px; /* Common ad width */
        min-height: 90px; /* Common ad height */
        display: flex;
        align-items: center;
        justify-content: center;
        font-size: 0.9rem;
        border-radius: 5px;
    }
    .ad-placeholder-sidebar { /* Example for a different type of ad */
```

```
            max-width: 300px;
            min-height: 250px;
        }




        main {
            max-width: 1200px;
            margin: 0 auto;
            padding: 0 1rem;
        }


        .tool-grid {
            display: grid;
            grid-template-columns: repeat(3, 1fr);
            gap: 1.5rem;
        }


        .tool-card {
            background-color: var(--tool-card-background);
            padding: 1.5rem;
            border-radius: 10px;
            box-shadow: var(--soft-box-shadow);
            transition: transform 0.3s ease, background-color 0.3s ease, color 0.3s ease,
    box-shadow 0.3s ease;
            display: flex;
            flex-direction: column;
            justify-content: space-between;
            opacity: 0; /* For fade-in animation */
            transform: translateY(20px); /* For fade-in animation */
            animation: fadeInUp 0.5s ease-out forwards;
        }


        .tool-card:hover {
            transform: translateY(-5px);
            background-color: var(--accent-color);
            color: var(--background-color);
            box-shadow: var(--card-hover-box-shadow);
        }


        .tool-card h2 {
            font-size: 1.5rem;
            margin-bottom: 0.75rem;
        color: var(--text-color);
        transition: color 0.3s ease;
}


.tool-card:hover h2 {
        color: var(--background-color);
}


.tool-card p {
        font-size: 0.9rem;
        margin-bottom: 1rem;
        flex-grow: 1;
        opacity: 0.9;
}


.tool-card:hover p {
        color: var(--background-color);
        opacity: 1;
}


.tool-button {
        background-color: var(--accent-color);
        color: var(--background-color);
        border: none;
        padding: 0.75rem 1.25rem;
        border-radius: 5px;
        font-weight: bold;
        cursor: pointer;
        text-align: center;
        transition: background-color 0.3s ease, color 0.3s ease;
        align-self: flex-start;
}


.tool-card:hover .tool-button {
        background-color: var(--background-color);
        color: var(--accent-color);
}


.tool-button:hover {
        background-color: var(--button-hover-color) !important;
        color: var(--background-color) !important;
}
```

```
    .modal {
        display: none;
        position: fixed;
        z-index: 1000;
        left: 0;
        top: 0;
        width: 100%;
        height: 100%;
        overflow: auto;
        background-color: rgba(0, 0, 0, 0.6);
        animation: fadeIn 0.3s ease-in-out;
    }


    @keyframes fadeIn {
        from { opacity: 0; }
        to { o

pacity: 1; }
    }


    .modal-content {
        background-color: var(--header-background);
        margin: 5% auto;
        padding: 25px;
        border-radius: 10px;
        width: 90%;
        max-width: 700px;
        box-shadow: 0 10px 30px rgba(0, 0, 0, 0.3);
        position: relative;
        animation: slideIn 0.3s ease-in-out;
    }

    /* Modal Ad Placeholder - example, might be too intrusive */
    /* .modal-ad-placeholder { margin-top: 1rem; } */




    @keyframes slideIn {
        from { transform: translateY(-50px); opacity: 0; }
        to { transform: translateY(0); opacity: 1; }
    }
```

```
.close-button {
	color: var(--text-color);
	float: right;
	font-size: 28px;
	font-weight: bold;
	transition: color 0.2s ease;
}


.close-button:hover,
.close-button:focus {
	color: var(--accent-color);
	text-decoration: none;
	cursor: pointer;
}


.modal-body {
	margin-top: 1.5rem;
	color: var(--text-color);
}


.modal-body h3 {
	color: var(--accent-color);
	margin-bottom: 1rem;
}


.modal-body label {
	display: block;
	margin-bottom: 0.5rem;
	font-weight: 600;
}


.modal-body input[type="text"],
.modal-body input[type="number"],
.modal-body input[type="date"],
.modal-body input[type="file"],
.modal-body textarea,
.modal-body select {
	width: 100%;
	padding: 0.75rem;
	margin-bottom: 1rem;
	border-radius: 5px;
	border: 1px solid var(--tool-card-background);
	background-color: var(--background-color);
```

```css
	color: var(--text-color);
	font-family: var(--font-family);
	font-size: 1rem;
}


.modal-body input[type="file"] { padding: 0.5rem; }
.modal-body textarea { min-height: 100px; resize: vertical; }


.modal-body button, .tool-interface button {
	background-color: var(--accent-color);
	color: var(--background-color);
	border: none;
	padding: 0.8rem 1.5rem;
	border-radius: 5px;
	font-weight: bold;
	cursor: pointer;
	transition: background-color 0.3s ease;
	margin-top: 0.5rem;
	margin-right: 0.5rem;
}


.modal-body button:hover, .tool-interface button:hover {
	background-color: var(--button-hover-color);
}


.modal-body .result-area, .tool-interface .result-area {
	margin-top: 1rem;
	padding: 1rem;
	background-color: var(--background-color);
	border-radius: 5px;
	border: 1px solid var(--tool-card-background);
	white-space: pre-wrap;
	word-wrap: break-word;
}


.modal-body .option-group {
	margin-bottom: 1rem;
	padding: 0.75rem;
	border: 1px solid var(--tool-card-background);
	border-radius: 5px;
}
.modal-body .option-group label { display: inline-block; margin-right: 10px; }
```

```
        .modal-alert {
                margin-top: 1rem;
                padding: 0.75rem;
                border-radius: 5px;
                font-weight: bold;
        }
        .modal-alert.success { background-color: #28a745; color: white; }
        .modal-alert.error { background-color: #dc3545; color: white; }
        .modal-alert.info { background-color:

#17a2b8; color: white; }


        #qrCodeContainer canvas {
                display: block;
                margin: 1rem auto;
                border: 5px solid var(--accent-color);
        }


        #imagePreview, #croppedImagePreview, #imagePreviewComp {
                max-width: 100%;
                max-height: 300px;
                margin-top: 1rem;
                border: 1px solid var(--accent-color);
                background-color: var(--background-color); /* Ensure preview area has
background */
        }

        #cropCanvas { /* Style for the cropper canvas */
                border:1px solid var(--accent-color);
                max-width:100%;
                display:block; /* Important for canvas */
                background-color: var(--background-color);
                margin-top: 10px;
        }




        .color-picker-display { display: flex; align-items: center; margin-top: 1rem; }
        .color-picker-display .color-preview { width: 50px; height: 50px; border-radius: 5px;
margin-right: 1rem; border: 1px solid var(--text-color); }
        .color-picker-display .color-values p { margin: 0.25rem 0; }
```

```
	.timer-display, .stopwatch-display { font-size: 2.5rem; font-weight: bold; text-align: center; margin-bottom: 1rem; color: var(--accent-color); }
	.laps-list { max-height: 150px; overflow-y: auto; border: 1px solid var(--tool-card-background); padding: 0.5rem; margin-top: 1rem; }
	.laps-list li { list-style: none; padding: 0.25rem 0; border-bottom: 1px dashed var(--tool-card-background); }
	.laps-list li:last-child { border-bottom: none; }

	.tabs { display:flex; margin-bottom:1rem; border-bottom:1px solid var(--tool-card-background); }
	.tab-button { background:none; border:none; color: var(--text-color); padding: 0.75rem 1rem; cursor:pointer; font-size: 1rem; font-family: var(--font-family); transition: background-color 0.3s, color 0.3s; }
	.tab-button.active { background-color: var(--accent-color); color: var(--background-color); border-bottom: 2px solid var(--accent-color); font-weight: bold; }
	.tab-button:not(.active):hover { background-color: var(--tool-card-background); }


	#toolsArticle {
		margin-top: 3rem;
		padding: 1.5rem;
		background-color: var(--tool-card-background);
		border-radius: 8px;
	}
	#toolsArticle h2 {
		color: var(--accent-color);
		margin-bottom: 1rem;
		text-align: center;
		font-size: 2rem;
	}
	#toolsArticle h3 {
		color: var(--accent-color);
		opacity: 0.9;
		margin-top: 1.5rem;
		margin-bottom: 0.5rem;
		border-bottom: 1px solid var(--header-background);
		padding-bottom: 0.3rem;
	}
	#toolsArticle p, #toolsArticle ul {
		margin-bottom: 1rem;
		font-size: 0.95rem;
		opacity: 0.9;
	}
	#toolsArticle li {
		margin-left: 1.5rem;
		margin-bottom: 0.5rem;
	}
	#toolsArticle strong { color: var(--accent-color); opacity: 1;}
```

```
    @keyframes fadeInUp {
        to { opacity: 1; transform: translateY(0); }
    }


    @media (max-width: 992px) { /* Tablet */
        .tool-grid { grid-template-columns: repeat(2, 1fr); }
        header h1 { font-size: 2rem; }
    }


    @media (max-width: 768px) { /* Mobile */
        .tool-grid { grid-template-columns: 1fr; }
        .modal-content { width: 95%; margin: 10% auto; padding: 20px; }
        header h1 { font-size: 1.8rem; }
        .tool-card h2 { font-size: 1.3rem; }
        .modal-body button, .tool-interface button { padding: 0.7rem 1.2rem; font-size: 0.9rem; }
        #toolsArticle h2 {

font-size: 1.6rem; }
        #toolsArticle h3 { font-size: 1.3rem; }
    }
    </style>
</head>
<body>
    <header>
    <h1>One Page Tools</h1>
    </header>


    <!-- Ad Placeholder 1 (Below Header) -->
    <div class="ad-placeholder">
    Your Advertisement Here (e.g., 728x90 Leaderboard)
    </div>


    <main>
    <div class="tool-grid">
        <!-- Tool cards will be dynamically generated by JavaScript -->
    </div>
    </main>
```

```
        <!-- Generic Modal Structure -->
        <div id="toolModal" class="modal">
        <div class="modal-content">
                <span class="close-button">&times;</span>
                <h2 id="modalTitle">Tool Title</h2>
                <div id="modalBody" class="modal-body">
                <!-- Tool-specific content will be injected here -->
                </div>
                <!-- Optional Ad Placeholder inside Modal -->
                <!-- <div class="ad-placeholder modal-ad-placeholder">Ad inside modal (e.g.,
300x250)</div> -->
                <div id="modalAlert" class="modal-alert" style="display:none;"></div>
        </div>
        </div>


        <!-- Article Section -->
        <div id="toolsArticleContainer" style="max-width: 1200px; margin: 2rem auto;
padding: 0 1rem;">
        <article id="toolsArticle">
                <h2>Welcome to One Page Tools: Your Ultimate Online Toolkit</h2>
                <p>In today's fast-paced digital world, efficiency and convenience are
paramount. Whether you're a student, a professional, a content creator, or simply someone
who interacts with digital media and data regularly, having quick access to a versatile set of
tools can be a game-changer. That's where <strong>One Page Tools</strong> comes in.
We've curated a collection of 20 essential, free-to-use online tools designed to simplify your
tasks, boost your productivity, and empower your digital endeavors – all from the comfort of
your web browser, with no installations required and a strong focus on client-side processing
for speed and privacy.</p>
                <p>Our platform is built with a modern, premium, and minimalistic design,
ensuring a seamless and enjoyable user experience across all devices – desktop, tablet,
and mobile. Each tool is crafted to be fully functional using only HTML, CSS, and Vanilla
JavaScript, meaning they are lightweight, fast, and respect your privacy by processing data
directly in your browser whenever possible.</p>

                <h3>Why Choose One Page Tools?</h3>
                <ul>
                <li><strong>Completely Free:</strong> All 20 tools are available at no
cost.</li>
                <li><strong>Client-Side Processing:</strong> For many tools, especially
those handling sensitive data or files, processing happens directly in your browser. This
means your files often don't even need to be uploaded to a server, enhancing speed and
privacy.</li>
                <li><strong>No Registration Required:</strong> Jump right in and start using
the tools without any sign-ups or logins.</li>
                <li><strong>User-Friendly Interface:</strong> Our clean, minimalistic design
makes it easy to find and use the tool you need.</li>
```

```html
			<li><strong>Fully Responsive:</strong> Access One Page Tools on any device, anytime, anywhere.</li>
			<li><strong>Diverse Range of Utilities:</strong> From image manipulation and media conversion to financial calculators and text utilities, we've got you covered.</li>
			</ul>


			<h3>Explore Our Suite of 20 Powerful Tools:</h3>


			<section id="article-image-tools">
			<h3>🎨 Image Manipulation Tools</h3>
			<p>Perfect for designers, photographers, and anyone working with digital images.</p>

			<h4>1. Image Converter</h4>
			<p><strong>What it does:</strong> Easily convert your images between popular form

ats like JPG, PNG, and WEBP. Whether you need a transparent background (PNG), a highly compressed web-friendly format (WEBP), or a standard photographic format (JPG), this tool handles it swiftly using browser-native canvas technology.</p>
			<p><strong>Use cases:</strong> Preparing images for web use, changing formats for compatibility, optimizing image types for specific needs.</p>


			<h4>2. Image Compressor</h4>
			<p><strong>What it does:</strong> Reduce the file size of your images without significant loss of quality. This tool uses canvas and adjustable quality settings (for JPEGs) to make your images lighter, faster to load on websites, and easier to share.</p>
			<p><strong>Use cases:</strong> Optimizing website images for faster loading times, reducing storage space, sending images via email or messaging apps with size limits.</p>


			<h4>3. Image Cropper</h4>
			<p><strong>What it does:</strong> Upload an image, define your desired crop area with a visual preview, and export the perfectly cropped section. Ideal for focusing on specific parts of an image or adjusting aspect ratios.</p>
			<p><strong>Use cases:</strong> Creating profile pictures, framing subjects, removing unwanted parts of an image, preparing images for specific layouts.</p>
			</section>


			<section id="article-media-tools">
			<h3>🎬 Audio & Video Tools</h3>
			<p>For content creators, students, or anyone needing to tweak audio and video files.</p>
			<h4>4. Video Converter (Browser-Limited)</h4>
			<p><strong>What it does:</strong> Convert video formats (e.g., MP4 to WebM or vice-versa) by re-encoding the video directly in your browser using MediaRecorder or canvas capabilities. Functionality and output quality depend on your browser's support for these technologies.</p>
			<p><strong>Use cases:</strong> Making videos compatible with specific platforms or devices, converting to web-friendly formats. (Note: Browser capabilities are key here).</p>


			<h4>5. Audio Converter (to WAV)</h4>
			<p><strong>What it does:</strong> Convert various audio files that your browser can play into the versatile WAV format. This tool utilizes the Web Audio API for decoding and processing.</p>
			<p><strong>Use cases:</strong> Preparing audio for editing software that prefers WAV, standardizing audio formats, creating uncompressed audio clips.</p>


			<h4>6. Audio Trimmer</h4>
			<p><strong>What it does:</strong> Upload an audio file, specify start and end times, and export the trimmed segment as a new WAV clip. Perfect for extracting soundbites, creating ringtones, or shortening audio recordings.</p>
			<p><strong>Use cases:</strong> Editing podcasts or interviews, creating short audio samples, removing unwanted sections from recordings.</p>
			</section>


			<section id="article-calculator-tools">
			<h3>🧮 Productivity & Financial Calculators</h3>
			<p>Essential tools for everyday calculations, financial planning, and health monitoring.</p>


			<h4>7. Age Calculator</h4>
			<p><strong>What it does:</strong> Simply input a date of birth, and this tool will instantly calculate the age in years, months, and days. Accurate and quick.</p>
			<p><strong>Use cases:</strong> Finding exact age for forms or records, curiosity, event planning based on age milestones.</p>


			<h4>8. EMI Calculator</h4>
			<p><strong>What it does:</strong> Plan your loans effectively. Input the loan amount, annual interest rate, and loan duration (in months) to calculate your Equated Monthly Installment (EMI), total interest payable, and total payment.</p>
			<p><strong>Use cases:</strong>
```

```
Financial planning for home loans, car loans, personal loans; understanding loan affordability.</p>


            <h4>9. SIP Calculator</h4>
            <p><strong>What it does:</strong> Estimate the future value of your Systematic Investment Plan (SIP) investments. Enter your monthly investment amount, expected annual interest rate, and investment duration (in years) to see potential returns.</p>
            <p><strong>Use cases:</strong> Mutual fund investment planning, retirement planning, goal-based savings projections.</p>


            <h4>10. BMI Calculator</h4>
            <p><strong>What it does:</strong> Calculate your Body Mass Index (BMI) by entering your weight (in kg) and height (in cm). The tool also provides your BMI category (e.g., Underweight, Normal, Overweight).</p>
            <p><strong>Use cases:</strong> Health and fitness tracking, understanding weight status, a general indicator for health assessment (consult a professional for medical advice).</p>
            </section>


            <section id="article-text-code-tools">
            <h3>✒️ Text & Code Utilities</h3>
            <p>A must-have for writers, developers, and anyone dealing with text or data.</p>


            <h4>11. QR Code Generator</h4>
            <p><strong>What it does:</strong> Enter any text, URL, contact information, or Wi-Fi credentials, and generate a downloadable QR code image instantly. Uses canvas for generation.</p>
            <p><strong>Use cases:</strong> Sharing website links, contact details, Wi-Fi access easily; marketing materials, event ticketing.</p>


            <h4>12. Password Generator</h4>
            <p><strong>What it does:</strong> Create strong, secure, and random passwords. Customize the length and include/exclude uppercase letters, lowercase letters, numbers, and symbols to meet various security requirements.</p>
            <p><strong>Use cases:</strong> Enhancing online security, creating unique passwords for different accounts, adhering to password complexity rules.</p>


            <h4>13. Word Counter</h4>
```

```
			<p><strong>What it does:</strong> A real-time text analysis tool. As you type or paste text, it counts words, characters (with and without spaces), spaces, and even estimates reading time. Essential for writers, students, and social media managers.</p>
			<p><strong>Use cases:</strong> Meeting word count requirements for essays or articles, optimizing social media posts, tracking writing progress.</p>


			<h4>14. Base64 Encoder/Decoder</h4>
			<p><strong>What it does:</strong> Easily convert plain text to Base64 encoding or decode Base64 strings back to their original plain text form. Useful for data transmission or simple obfuscation.</p>
			<p><strong>Use cases:</strong> Encoding data for URLs or HTML, decoding data from various web sources, simple data protection (not encryption).</p>


			<h4>15. JSON Formatter</h4>
			<p><strong>What it does:</strong> Paste your JSON data, and this tool will auto-format (pretty-print) it with proper indentation and syntax highlighting. It also helps validate the JSON structure and points out errors.</p>
			<p><strong>Use cases:</strong> Debugging JSON APIs, making complex JSON data readable, validating JSON structures before use in applications.</p>
			</section>

			<section id="article-accessibility-interaction-tools">
			<h3>🗣️ Accessibility & Interactive Tools</h3>
			<p>Enhance accessibility and engage with content in new ways.</p>


			<h4>16. Color Picker Tool</h4>
			<p><strong>What it does:</strong> Visually pick a color using a standard color input, or enter a color code. The tool displays its HEX, RGB, and HSL values, making it easy to work with colors for web design

, graphic design, or any digital project.</p>
			<p><strong>Use cases:</strong> Web development, graphic design, digital art, matching colors, understanding color codes.</p>


			<h4>17. Text to Speech</h4>
			<p><strong>What it does:</strong> Enter any text, and listen to it being read aloud by your browser's built-in speech synthesis engine. You can often choose from different voices, and adjust the rate and pitch.</p>
			<p><strong>Use cases:</strong> Accessibility for visually impaired users, proofreading text, learning pronunciation, multitasking by listening to content.</p>


			<h4>18. Speech to Text</h4>
```

```html
			<p><strong>What it does:</strong> Use your microphone to convert your spoken words into text. This tool leverages your browser's Web Speech API for real-time voice recognition.</p>
			<p><strong>Use cases:</strong> Dictating notes, writing emails hands-free, accessibility for users with motor impairments, transcribing short audio clips.</p>
			</section>


			<section id="article-measurement-time-tools">
			<h3>📏 Measurement & Time Management Tools</h3>
			<p>Practical utilities for conversions and keeping track of time.</p>


			<h4>19. Unit Converter</h4>
			<p><strong>What it does:</strong> A versatile converter for various units of measurement, including length (meters, feet, miles, etc.), weight (kilograms, pounds, ounces, etc.), and temperature (Celsius, Fahrenheit, Kelvin). Simple and intuitive.</p>
			<p><strong>Use cases:</strong> Converting recipes, scientific calculations, travel planning, everyday measurement conversions.</p>


			<h4>20. Timer / Stopwatch Tool</h4>
			<p><strong>What it does:</strong> A dual-function tool. Set a countdown timer for tasks or use the stopwatch with lap functionality to measure elapsed time accurately. Clean interface and easy controls.</p>
			<p><strong>Use cases:</strong> Time management (Pomodoro technique), cooking, workouts, tracking time for activities, conducting experiments.</p>
			</section>


			<h3>Privacy and Security at the Forefront</h3>
			<p>We understand the importance of your data. That's why One Page Tools is designed with a "privacy-first" approach. Many of our tools, particularly those that handle files (like image and audio converters/compressors/croppers), perform all operations directly within your browser. This means your files are not uploaded to our servers, offering you enhanced security and speed. For tools that require input (like calculators or text utilities), the data is processed client-side and is not stored or tracked beyond what's necessary for the tool's immediate function or basic anonymous usage analytics (if enabled by you).</p>


			<h3>Constantly Evolving</h3>
			<p>One Page Tools is a living project. We are committed to maintaining and improving our existing tools and potentially adding new ones based on user needs and technological advancements. Our goal is to be your go-to destination for quick, reliable, and free online utilities.</p>
			<p>We invite you to explore the One Page Tools and discover how our suite of tools can simplify your digital life. Bookmark us and share with friends and colleagues who might benefit from these handy utilities!</p>
```

```
	</article>
	</div>


	<!-- Ad Placeholder 2 (Before Footer/End of Page) -->
	<div class="ad-placeholder">
	Your Advertisement Here (e.g., 728x90 or 300x250)
	</div>




	<script>
	// --- QR Code Generation Library (Minimal Placeholder - qrcode.js by davidshimjs is a good option to embed) ---
	// For a fully functional QR code, you would embed a library like qrcode.js here.
	// Example structure if qrcode.js was embedded:
	/

*
	var QRCode; (function(){ // QRCode Kapselung ... entire qrcode.js library code ... QRCode=t})()
	*/
	// Since embedding a full library makes this example overly long, the QR tool uses a simplified canvas drawing.
	// A real implementation should use a proper QR library for compliant codes.
	// For demonstration, if you had a qrcode.js file, you'd paste its contents here.
	// For now, the script will check for a 'QRCode' object and use a fallback if not found.




	document.addEventListener('DOMContentLoaded', () => {
		const toolGrid = document.querySelector('.tool-grid');
		const modal = document.getElementById('toolModal');
		const modalTitle = document.getElementById('modalTitle');
		const modalBody = document.getElementById('modalBody');
		const modalAlert = document.getElementById('modalAlert');
		const closeButton = document.querySelector('.close-button');


		// --- Basic Browser-Native Analytics (Tool Usage Logger) ---
		function logToolUsage(toolId) {
		try {
		let usageData = JSON.parse(localStorage.getItem('multiToolHubUsage')) || [];
		usageData.push({ tool: toolId, timestamp: new Date().toISOString() });
		// Optional: Limit the size of usageData to prevent localStorage overflow
		if (usageData.length > 1000) { // Keep last 1000 entries
```

```
                usageData = usageData.slice(usageData.length - 1000);
        }
        localStorage.setItem('multiToolHubUsage', JSON.stringify(usageData));
        } catch (e) {
        console.error("Error logging tool usage:", e);
        }
        }
        // Example: To see logged data, open browser console and type:
        // JSON.parse(localStorage.getItem('multiToolHubUsage'))



        const tools = [
        { id: 'imageConverter', title: 'Image Converter', description: 'Convert between
JPG, PNG, and WEBP formats.' },
        { id: 'imageCompressor', title: 'Image Compressor', description: 'Compress
image file size with quality settings.' },
        { id: 'imageCropper', title: 'Image Cropper', description: 'Upload, crop image
with preview, and export.' },
        { id: 'videoConverter', title: 'Video Converter', description: 'Convert video (MP4
↔ WebM via re-encoding). Browser-limited.' },
        { id: 'audioConverter', title: 'Audio Converter', description: 'Convert uploaded
audio to WAV format.' },
        { id: 'audioTrimmer', title: 'Audio Trimmer', description: 'Upload, trim audio,
and export trimmed WAV clip.' },
        { id: 'ageCalculator', title: 'Age Calculator', description: 'Calculate age from
date of birth.' },
        { id: 'emiCalculator', title: 'EMI Calculator', description: 'Calculate Equated
Monthly Installment for loans.' },
        { id: 'sipCalculator', title: 'SIP Calculator', description: 'Calculate future value of
SIP investments.' },
        { id: 'qrCodeGenerator', title: 'QR Code Generator', description: 'Generate
downloadable QR codes from text/URL.' },
        { id: 'passwordGenerator', title: 'Password Generator', description: 'Generate
secure passwords with custom options.' },
        { id: 'wordCounter', title: 'Word Counter', description: 'Count words,
characters, spaces, and estimate reading time.' },
        { id: 'base64EncoderDecoder', title: 'Base64 Encoder/Decoder', description:
'Encode to Base64 or decode from Base64.' },
        { id: 'colorPicker', title: 'Color Picker Tool', description: 'Pick colors and get
HEX, RGB, HSL values.' },
        { id: 'textToSpeech', title: 'Text to Speech', description: 'Convert text into
spoken audio

using browser voices.' },
        { id: 'speechToText', title: 'Speech to Text', description: 'Convert voice from
microphone into text.' },
        { id: 'jsonFormatter', title: 'JSON Formatter', description: 'Format and validate JSON data.' },
        { id: 'unitConverter', title: 'Unit Converter', description: 'Convert values between various units (length, weight, temp).' },
        { id: 'bmiCalculator', title: 'BMI Calculator', description: 'Calculate Body Mass Index and category.' },
        { id: 'timerStopwatch', title: 'Timer / Stopwatch', description: 'Simple timer and stopwatch functionality.' },
        ];


        function openModal(tool) {
        modalTitle.textContent = tool.title;
        modalBody.innerHTML = '';
        hideAlert();

        const toolFunction = toolInitializers[tool.id];
        if (toolFunction) {
        toolFunction(modalBody);
        logToolUsage(tool.id); // Log tool opening
        } else {
        modalBody.innerHTML = '<p>Tool UI not implemented yet.</p>';
        }
        modal.style.display = 'block';
        }


        function closeModal() {
        modal.style.display = 'none';
        modalBody.innerHTML = '';
        if (window.currentToolCleanup) {
        window.currentToolCleanup();
        window.currentToolCleanup = null;
        }
        }


        closeButton.onclick = closeModal;
        window.onclick = (event) => {
        if (event.target === modal) {
        closeModal();
        }
        };

        function showAlert(message, type = 'info') {
        modalAlert.textContent = message;
        modalAlert.className = `modal-alert ${type}`;
        modalAlert.style.display = 'block';
```

```
        }


        function hideAlert() {
        modalAlert.style.display = 'none';
        }


        tools.forEach((tool, index) => {
        const card = document.createElement('div');
        card.className = 'tool-card';
        card.style.animationDelay = `${index * 0.05}s`;
        card.innerHTML = `
        <h2>${tool.title}</h2>
        <p>${tool.description}</p>
        <button class="tool-button" data-toolid="${tool.id}">Open Tool</button>
        `;
        card.querySelector('.tool-button').addEventListener('click', () =>
openModal(tool));
        toolGrid.appendChild(card);
        });


        const toolInitializers = {
        imageConverter: (container) => {
        container.innerHTML = `
            <input type="file" id="imgConvFile"
accept="image/jpeg,image/png,image/webp">
            <label for="imgConvFormat">Convert to:</label>
            <select id="imgConvFormat">
            <option value="image/jpeg">JPEG</option>
            <option value="image/png">PNG</option>
            <option value="image/webp">WEBP</option>
            </select>
            <button id="imgConvButton">Convert & Download</button>
            <p>Note: WEBP support varies by browser.</p>
            <img id="imagePreview" src="#" alt="Preview" style="display:none;">
        `;
        const fileInput = container.querySelector('#imgConvFile');
        const formatSelect = container.querySelector('#imgConvFormat');
        const convertButton = container.querySelect

or('#imgConvButton');
        const preview = container.querySelector('#imagePreview');
        let originalFileName = 'converted_image';


        fileInput.onchange = (e) => {
            if (e.target.files && e.target.files[0]) {
            originalFileName = e.target.files[0].name.split('.')[0] || 'image';
            const reader = new FileReader();
            reader.onload = (event) => {
            preview.src = event.target.result;
            preview.style.display = 'block';
            }
            reader.readAsDataURL(e.target.files[0]);
            } else {
            preview.style.display = 'none';
            preview.src="#";
            }
        };


        convertButton.onclick = () => {
            if (!fileInput.files || fileInput.files.length === 0) {
            showAlert('Please select an image file first.', 'error');
            return;
            }
            const file = fileInput.files[0];
            const targetFormat = formatSelect.value;
            const targetExtension = targetFormat.split('/')[1];


            showAlert('Processing...', 'info');


            const reader = new FileReader();
            reader.onload = (event) => {
            const img = new Image();
            img.onload = () => {
            const canvas = document.createElement('canvas');
            canvas.width = img.width;
            canvas.height = img.height;
            const ctx = canvas.getContext('2d');
            ctx.drawImage(img, 0, 0);

            canvas.toBlob((blob) => {
                if (blob) {
                const url = URL.createObjectURL(blob);
                const a = document.createElement('a');
                a.href = url;
                a.download =
`${originalFileName}_converted.${targetExtension}`;
                document.body.appendChild(a);
                a.click();
                document.body.removeChild(a);
                URL.revokeObjectURL(url);
                showAlert('Conversion successful!', 'success');
                } else {
                showAlert(`Error converting to ${targetFormat}. This format
might not be supported for export by your browser.`, 'error');
                }
            }, targetFormat, 0.9);
            };
            img.onerror = () => showAlert('Could not load image. Ensure it is a
valid JPG, PNG, or WEBP.', 'error');
            img.src = event.target.result;
            };
            reader.readAsDataURL(file);
        };
        },


        imageCompressor: (container) => {
        container.innerHTML = `
            <input type="file" id="imgCompFile" accept="image/jpeg,image/png">
            <label for="imgCompQuality">Quality (0.1 - 1.0 for JPEG, ignored for
PNG):</label>
            <input type="number" id="imgCompQuality" value="0.7" min="0.1"
max="1.0" step="0.1">

<button id="imgCompButton">Compress & Download</button>
            <p>PNG compression is lossless and quality setting is ignored.</p>
            <div id="compressionInfo" class="result-area"
style="display:none;"></div>
            <img id="imagePreviewComp" src="#" alt="Preview"
style="display:none;">
        `;
        const fileInput = container.querySelector('#imgCompFile');
        const qualityInput = container.querySelector('#imgCompQuality');
        const compressButton = container.querySelector('#imgCompButton');
        const compressionInfo = container.querySelector('#compressionInfo');
        const preview = container.querySelector('#imagePreviewComp');
        let originalFileName = 'compressed_image';


        fileInput.onchange = (e) => {
            if (e.target.files && e.target.files[0]) {
            originalFileName = e.target.files[0].name.split('.')[0] || 'image';
            const reader = new FileReader();
            reader.onload = (event) => {
            preview.src = event.target.result;
            preview.style.display = 'block';
            }
            reader.readAsDataURL(e.target.files[0]);
            } else {
            preview.style.display = 'none';
            preview.src = "#";
            }
        };


        compressButton.onclick = () => {
            if (!fileInput.files || fileInput.files.length === 0) {
            showAlert('Please select an image file.', 'error');
            return;
            }
            const file = fileInput.files[0];
            const quality = parseFloat(qualityInput.value);
            const originalSize = (file.size / 1024).toFixed(2);
            showAlert('Processing...', 'info');


            const reader = new FileReader();
            reader.onload = (event) => {
            const img = new Image();
            img.onload = () => {
            const canvas = document.createElement('canvas');
            const ctx = canvas.getContext('2d');
            canvas.width = img.width;
            canvas.height = img.height;
            ctx.drawImage(img, 0, 0);


            let outputFormat = file.type === 'image/png' ? 'image/png' :
'image/jpeg';
            let extension = outputFormat.split('/')[1];


            canvas.toBlob((blob) => {
                if (blob) {
                const compressedSize = (blob.size / 1024).toFixed(2);
                const url = URL.createObjectURL(blob);
                const a = document.createElement('a');
                a.href = url;
                a.download = `${originalFileName}_compressed.${extension}`;
                document.body.appendChild(a);
                a.click();
                document.body.removeChild(a);
                URL.revokeObjectURL(url);
                compressionInfo.innerHTML = `Original Size: ${originalSize}
KB<br>Compressed Size: ${compressedSize} KB<br>Reduction: ${(

(1 - compressedSize / originalSize) * 100).toFixed(2)}%`;
                compressionInfo.style.display = 'block';
                showAlert('Compression successful!', 'success');
                } else {
                showAlert('Error during compression.', 'error');
                }
            }, outputFormat, outputFormat === 'image/jpeg' ? quality : undefined);
            };
            img.onerror = () => showAlert('Could not load image.', 'error');
            img.src = event.target.result;
            };
            reader.readAsDataURL(file);
        };
        },


        imageCropper: (container) => {
        container.innerHTML = `
            <input type="file" id="imgCropFile" accept="image/*">
            <p>After selecting an image, adjust crop parameters below. A real
cropper would have a draggable overlay.</p>
            <canvas id="cropCanvas" style="display:none;"></canvas>
            <br>
            <label for="cropX">Crop Start X (px):</label> <input type="number"
id="cropX" value="0" style="width:80px;">
            <label for="cropY">Crop Start Y (px):</label> <input type="number"
id="cropY" value="0" style="width:80px;">
            <br>
            <label for="cropW">Crop Width (px):</label> <input type="number"
id="cropW" value="100" style="width:80px;">
            <label for="cropH">Crop Height (px):</label> <input type="number"
id="cropH" value="100" style="width:80px;">
            <br>
            <button id="cropButton">Crop & Download</button>
            <img id="croppedImagePreview" src="#" alt="Cropped Preview"
style="display:none;">
        `;
        const fileInput = container.querySelector('#imgCropFile');
        const cropCanvas = container.querySelector('#cropCanvas'); // This is the
preview canvas
        const ctx = cropCanvas.getContext('2d');
        const cropXInput = container.querySelector('#cropX');
        const cropYInput = container.querySelector('#cropY');
        const cropWInput = container.querySelector('#cropW');
        const cropHInput = container.querySelector('#cropH');
        const cropButton = container.querySelector('#cropButton');
        const croppedPreview = container.querySelector('#croppedImagePreview');
        let sourceImage = null;
        let originalFileName = 'cropped_image';


        fileInput.onchange = (e) => {
            if (e.target.files && e.target.files[0]) {
            originalFileName = e.target.files[0].name.split('.')[0] || 'image';
            const reader = new FileReader();
            reader.onload = (event) => {
            sourceImage = new Image();
            sourceImage.onload = () => {
                const MAX_PREVIEW_DIM = 300; // Max width/height for
preview canvas
                let scale = Math.min(MAX_PREVIEW_DIM /
sourceImage.width, MAX_PREVIEW_DIM / sourceImage.height, 1);
                cropCanvas.width = sourceImage.width * scale;
                cropCanvas.height = sourceImage.height * scale;
                ctx.drawImage(sourceImage, 0, 0, cropCanvas.width,
cropCanvas.height);
                cropCanvas.style.display = 'block';

cropWInput.value = Math.floor(sourceImage.width / 2);
                cropHInput.value = Math.floor(sourceImage.height / 2);
                cropXInput.value = Math.floor(sourceImage.width / 4);
                cropYInput.value = Math.floor(sourceImage.height / 4);
                showAlert('Image loaded. Adjust crop parameters based on
original image dimensions.', 'info');
            };
            sourceImage.src = event.target.result;
            };
            reader.readAsDataURL(e.target.files[0]);
            } else {
            cropCanvas.style.display = 'none';
            if (sourceImage) sourceImage.src = "";
            sourceImage = null;
            }
        };


        cropButton.onclick = () => {
            if (!sourceImage) {
            showAlert('Please select an image first.', 'error');
            return;
            }
            const sx = parseInt(cropXInput.value);
            const sy = parseInt(cropYInput.value);
            const sWidth = parseInt(cropWInput.value);
            const sHeight = parseInt(cropHInput.value);


            if (isNaN(sx) || isNaN(sy) || isNaN(sWidth) || isNaN(sHeight) || sWidth
<= 0 || sHeight <= 0) {
            showAlert('Invalid crop dimensions. Ensure they are positive
numbers.', 'error');
            return;
            }
            if (sx + sWidth > sourceImage.width || sy + sHeight >
sourceImage.height || sx < 0 || sy < 0) {
            showAlert('Crop area is outside the image boundaries.', 'error');
            return;
            }

            showAlert('Cropping...', 'info');
            const tempCanvas = document.createElement('canvas');
            tempCanvas.width = sWidth;
            tempCanvas.height = sHeight;
            const tempCtx = tempCanvas.getContext('2d');

            try {
            tempCtx.drawImage(sourceImage, sx, sy, sWidth, sHeight, 0, 0,
sWidth, sHeight);
            const dataUrl = tempCanvas.toDataURL(fileInput.files[0].type ||
'image/png');
            croppedPreview.src = dataUrl;
            croppedPreview.style.display = 'block';


            const a = document.createElement('a');
            a.href = dataUrl;
            a.download = `${originalFileName}_cropped.${(fileInput.files[0].type ||
'image/png').split('/')[1]}`;
            document.body.appendChild(a);
            a.click();
            document.body.removeChild(a);
            showAlert('Image cropped and download started.', 'success');
            } catch (error) {
            showAlert('Error during cropping: ' + error.message, 'error');
            }
        };
        },


        videoConverter: (container) => {
        container.innerHTML = `
```

```
            <p>This tool attempts to convert playable videos to MP4 or WebM by
re-encoding using MediaRecorder. Success and output quality depend on browser
capabilities.</p>
            <input type="file" id="vidConvFile" accept="video/*">

<label for="vidConvFormat">Convert to:</label>
            <select id="vidConvFormat">
            <option value="video/webm;codecs=vp8,opus">WebM
(VP8/Opus)</option>
            <option value="video/webm;codecs=vp9,opus">WebM (VP9/Opus -
better quality)</option>
            <option value="video/mp4;codecs=avc1.42E01E,mp4a.40.2">MP4
(H.264/AAC - browser support varies)</option>
            </select>
            <button id="vidConvButton">Convert & Download</button>
            <video id="vidConvPreview" controls style="max-width:100%;
margin-top:10px; display:none; background-color:black;"></video>
            <p id="vidConvStatus" class="result-area" style="display:none;"></p>
        `;
        const fileInput = container.querySelector('#vidConvFile');
        const formatSelect = container.querySelector('#vidConvFormat');
        const convertButton = container.querySelector('#vidConvButton');
        const videoPreview = container.querySelector('#vidConvPreview');
        const statusP = container.querySelector('#vidConvStatus');
        let mediaRecorder;
        let recordedChunks = [];
        let originalFileName = 'converted_video';


        fileInput.onchange = (e) => {
            recordedChunks = []; // Reset chunks
            if (mediaRecorder && mediaRecorder.state === "recording") {
            mediaRecorder.stop();
            }
            if (e.target.files && e.target.files[0]) {
            const file = e.target.files[0];
            originalFileName = file.name.split('.')[0] || 'video';
            const url = URL.createObjectURL(file);
            videoPreview.src = url;
            videoPreview.style.display = 'block';
            videoPreview.onloadedmetadata = () => {
            showAlert(`Video loaded. Duration:
${videoPreview.duration.toFixed(2)}s. Ready to convert.`, 'info');
            statusP.style.display = 'none';
            }
            videoPreview.onerror = () => showAlert('Error loading video. It might
be an unsupported format.', 'error');
            } else {
```

```
            videoPreview.style.display = 'none';
            videoPreview.src = "";
            }
        };


        convertButton.onclick = () => {
            if (!videoPreview.src || !videoPreview.src.startsWith('blob:')) {
            showAlert('Please select a video file first.', 'error');
            return;
            }


            const targetMimeType = formatSelect.value;
            if (!MediaRecorder.isTypeSupported(targetMimeType)) {
            showAlert(`Your browser does not support recording to
${targetMimeType}. Try another format or browser. Supported types often include
'video/webm', 'video/webm;codecs=vp8', 'video/webm;codecs=vp9'. MP4 support is less
common for MediaRecorder.`, 'error');
            return;
            }


            showAlert('Starting conversion... The video will play. Do not close
modal.', 'info');
            statusP.textContent = 'Conversion in progress... 0%';
            statusP.style.display = 'block';

            videoPreview.currentTime = 0;

            const stream = videoPreview.captureStre

am ? videoPreview.captureStream() : videoPreview.mozCaptureStream ?
videoPreview.mozCaptureStream() : null;


            if (!stream) {
            showAlert('Could not capture video stream. Your browser might not
support this feature (captureStream/mozCaptureStream).', 'error');
            return;
            }


            recordedChunks = [];
            try {
            mediaRecorder = new MediaRecorder(stream, { mimeType:
targetMimeType });
            } catch (err) {
```

```
                        showAlert(`Error initializing MediaRecorder with ${targetMimeType}:
${err.message}. Try a simpler WebM option.`, 'error');
                        return;
                        }




                        mediaRecorder.ondataavailable = (event) => {
                        if (event.data.size > 0) {
                        recordedChunks.push(event.data);
                        }
                        };


                        mediaRecorder.onstop = () => {
                        if (recordedChunks.length === 0) {
                        showAlert('Conversion stopped, but no data was recorded. The
selected format/codec might be problematic for your browser.', 'error');
                        statusP.textContent = 'Conversion failed: No data.';
                        return;
                        }
                        const blob = new Blob(recordedChunks, { type:
targetMimeType.split(';')[0] }); // Use base MIME type for Blob
                        const url = URL.createObjectURL(blob);
                        const a = document.createElement('a');
                        a.href = url;
                        a.download =
`${originalFileName}_converted.${targetMimeType.split('/')[1].split(';')[0]}`;
                        document.body.appendChild(a);
                        a.click();
                        document.body.removeChild(a);
                        URL.revokeObjectURL(url);
                        showAlert('Video conversion finished!', 'success');
                        statusP.textContent = 'Conversion complete!';
                        videoPreview.pause();
                        };

                        mediaRecorder.onerror = (event) => {
                        showAlert(`MediaRecorder error: ${event.error ? event.error.name :
'Unknown error'}. Try a different format.`, 'error');
                        statusP.textContent = `Error: ${event.error ? event.error.name :
'Unknown'}`;
                        videoPreview.pause();
                        };

                        videoPreview.onended = () => {
                        if (mediaRecorder && mediaRecorder.state === 'recording') {
```

```
            mediaRecorder.stop();
            }
            };

            videoPreview.ontimeupdate = () => {
            if (videoPreview.duration && mediaRecorder && mediaRecorder.state
=== 'recording') {
            const progress = (videoPreview.currentTime / videoPreview.duration) *
100;
            statusP.textContent = `Conversion in progress...
${progress.toFixed(0)}%`;
            }
            };


            videoPreview.play().then(() => {
            mediaRecorder.start(1000); // Timeslice: ondataavailable every 1s
(helps with large files)
            }).catch(err => {
            showAlert(`Error playing video for conversion: ${err.message}`, 'error');

statusP.textContent = `Error: ${err.message}`;
            });
        };
        window.currentToolCleanup = () => {
            if (mediaRecorder && mediaRecorder.state === 'recording') {
            mediaRecorder.stop();
            }
            if(videoPreview) videoPreview.pause();
            // Revoke object URL if videoPreview.src is a blob URL to free
memory
            if (videoPreview && videoPreview.src &&
videoPreview.src.startsWith('blob:')) {
            URL.revokeObjectURL(videoPreview.src);
            }
        };
        },

        audioConverter: (() => { // IIFE to share bufferToWave
        const audioContext = new (window.AudioContext ||
window.webkitAudioContext)();

        function bufferToWave(abuffer) { // No 'len' needed, use abuffer.length
            let numOfChan = abuffer.numberOfChannels,
            length = abuffer.length * numOfChan * 2 + 44, // Correct length
calculation
            buffer = new ArrayBuffer(length),
            view = new DataView(buffer),
            channels = [], i, sample,
            offset = 0, // Frame offset
            pos = 0;          // Byte offset in buffer

            // write WAVE header
            view.setUint32(pos, 0x46464952, false); pos += 4; // "RIFF"
            view.setUint32(pos, length - 8, true); pos += 4;  // file length - 8
            view.setUint32(pos, 0x45564157, false); pos += 4; // "WAVE"

            view.setUint32(pos, 0x20746d66, false); pos += 4; // "fmt " chunk
            view.setUint32(pos, 16, true); pos += 4;                  // length = 16
            view.setUint16(pos, 1, true); pos += 2;                   // PCM (uncompressed)
            view.setUint16(pos, numOfChan, true); pos += 2;
            view.setUint32(pos, abuffer.sampleRate, true); pos += 4;
            view.setUint32(pos, abuffer.sampleRate * 2 * numOfChan, true); pos += 4; // "byte rate"
            view.setUint16(pos, numOfChan * 2, true); pos += 2; // block align (channels * bytes/sample)
            view.setUint16(pos, 16, true); pos += 2;                  // bits per sample

            view.setUint32(pos, 0x61746164, false); pos += 4; // "data" - chunk
            view.setUint32(pos, abuffer.length * numOfChan * 2, true); pos += 4; // chunk length (audio data size)

            for (i = 0; i < abuffer.numberOfChannels; i++)
            channels.push(abuffer.getChannelData(i));

            // Write interleaved data
            for (offset = 0; offset < abuffer.length; offset++) {
            for (i = 0; i < numOfChan; i++) {
            sample = Math.max(-1, Math.min(1, channels[i][offset])); // clamp
            sample = sample < 0 ? sample * 0x8000 : sample * 0x7FFF; // scale to 16-bit signed int
            view.setInt16(pos, sample, true);
            pos += 2;
            }
            }
            return new Blob([view], { type: 'audio/wav' });
        }

        return (container) => { // This is the actual toolInitializer function
```

```
            container.innerHTML = `
            <p>This tool converts various audio formats (that your browser can
play) into WAV format.</p>

<input type="file" id="audioConvFile" accept="audio/*">
            <button id="audioConvButton">Convert to WAV & Download</button>
            <audio id="audioConvPreview" controls style="width:100%;
margin-top:10px; display:none;"></audio>
            `;
            const fileInput = container.querySelector('#audioConvFile');
            const convertButton = container.querySelector('#audioConvButton');
            const audioPreview = container.querySelector('#audioConvPreview');
            let decodedAudioBuffer = null; // Renamed to avoid conflict
            let originalFileName = 'converted_audio';



            fileInput.onchange = (e) => {
            if (e.target.files && e.target.files[0]) {
            const file = e.target.files[0];
            originalFileName = file.name.split('.')[0] || 'audio';
            showAlert('Loading audio...', 'info');
            const reader = new FileReader();
            reader.onload = (event) => {
                audioContext.decodeAudioData(event.target.result)
                .then(buffer => { // Changed var name here
                decodedAudioBuffer = buffer;
                // Create a temporary URL for preview to avoid issues with
large files / ArrayBuffer
                const tempUrl = URL.createObjectURL(file);
                audioPreview.src = tempUrl;
                audioPreview.style.display = 'block';
                audioPreview.oncanplaythrough = () =>
URL.revokeObjectURL(tempUrl); // Revoke after load
                showAlert('Audio loaded. Ready to convert to WAV.',
'success');
                })
                .catch(err => showAlert(`Error decoding audio: ${err.message}.
Try a different format.`, 'error'));
            };
            reader.onerror = () => showAlert('Error reading file.', 'error');
            reader.readAsArrayBuffer(file);
            } else {
            audioPreview.style.display = 'none';
            audioPreview.src = "";
            decodedAudioBuffer = null;
            }
            };
            convertButton.onclick = () => {
            if (!decodedAudioBuffer) {
            showAlert('Please select and load an audio file first.', 'error');
            return;
            }
            showAlert('Converting to WAV...', 'info');
            try {
            const wavBlob = bufferToWave(decodedAudioBuffer); // Use the
shared function
            const url = URL.createObjectURL(wavBlob);
            const a = document.createElement('a');
            a.href = url;
            a.download = `${originalFileName}_converted.wav`;
            document.body.appendChild(a);
            a.click();
            document.body.removeChild(a);
            URL.revokeObjectURL(url);
            showAlert('Conversion to WAV successful!', 'success');
            } catch (err) {

showAlert(`Error converting to WAV: ${err.message}`, 'error');
            }
            };
            // Make bufferToWave accessible for the trimmer
            toolInitializers.audioConverter.bufferToWave = bufferToWave;


            window.currentToolCleanup = () => {
            if (audioPreview && audioPreview.src &&
audioPreview.src.startsWith('blob:')) {
            URL.revokeObjectURL(audioPreview.src);
            }
            };
        };
        })(), // Immediately invoke the IIFE


        audioTrimmer: (container) => {
        const audioContext = new (window.AudioContext ||
window.webkitAudioContext)();
        container.innerHTML = `
            <input type="file" id="trimAudioFile" accept="audio/*">
            <audio id="trimAudioPreview" controls style="width:100%;
margin-top:10px; display:none;"></audio>
            <div id="trimControls" style="display:none; margin-top:10px;">
            <label for="trimStartTime">Start Time (s):</label>
            <input type="number" id="trimStartTime" value="0" min="0"
step="0.01">
            <label for="trimEndTime">End Time (s):</label>
            <input type="number" id="trimEndTime" value="0" min="0"
step="0.01">
            <p id="audioDurationInfo"></p>
            <button id="trimButton">Trim & Download WAV</button>
            </div>
        `;
        const fileInput = container.querySelector('#trimAudioFile');
        const audioPreview = container.querySelector('#trimAudioPreview');
        const trimControls = container.querySelector('#trimControls');
        const startTimeInput = container.querySelector('#trimStartTime');
        const endTimeInput = container.querySelector('#trimEndTime');
        const durationInfo = container.querySelector('#audioDurationInfo');
        const trimButton = container.querySelector('#trimButton');

        let sourceBuffer = null;
        let originalFileName = 'trimmed_audio';


        fileInput.onchange = (e) => {
            if (e.target.files && e.target.files[0]) {
            const file = e.target.files[0];
            originalFileName = file.name.split('.')[0] || 'audio';
            showAlert('Loading audio...', 'info');
            const reader = new FileReader();
            reader.onload = (event) => {
            audioContext.decodeAudioData(event.target.result)
                .then(decodedBuffer => {
                sourceBuffer = decodedBuffer;
                const tempUrl = URL.createObjectURL(file);
                audioPreview.src = tempUrl;
                audioPreview.style.display = 'block';
                audioPreview.onloadedmetadata = () => {
                URL.revokeObjectURL(tempUrl); // Revoke after duration is
known
                const duration = sourceBuffer.duration; // Use buffer duration
for accuracy
                durationInfo.textContent = `Duration: ${duration.toFixed(2)}s`;
                endTimeInput.value = duration.toFixed(2);
                endTimeInput.max = duration.toFixed(2);

startTimeInput.max = duration.toFixed(2);
                trimControls.style.display = 'block';
                showAlert('Audio loaded. Set trim times.', 'success');
                };
                audioPreview.onerror = () => showAlert('Error playing preview.', 'error');
                })
                .catch(err => showAlert(`Error decoding audio: ${err.message}`, 'error'));
            };
            reader.readAsArrayBuffer(file);
            } else {
            audioPreview.style.display = 'none';
            trimControls.style.display = 'none';
            audioPreview.src = "";
            sourceBuffer = null;
            }
        };


        trimButton.onclick = () => {
            if (!sourceBuffer) {
            showAlert('Please load an audio file first.', 'error');
            return;
            }
            const startTime = parseFloat(startTimeInput.value);
            const endTime = parseFloat(endTimeInput.value);


            if (isNaN(startTime) || isNaN(endTime) || startTime < 0 || endTime <= startTime || endTime > sourceBuffer.duration) {
            showAlert('Invalid start/end times. Ensure End Time is after Start Time and within audio duration.', 'error');
            return;
            }

            showAlert('Trimming audio...', 'info');
            try {
            const startOffset = Math.floor(startTime * sourceBuffer.sampleRate);
            const endOffset = Math.floor(endTime * sourceBuffer.sampleRate);
            const frameCount = endOffset - startOffset;


            if (frameCount <=0) {
            showAlert('Trimmed duration is zero or negative.', 'error');
            return;
            }


            const trimmedBuffer = audioContext.createBuffer(
            sourceBuffer.numberOfChannels,
            frameCount,
            sourceBuffer.sampleRate
            );


            for (let i = 0; i < sourceBuffer.numberOfChannels; i++) {
            const channelData = sourceBuffer.getChannelData(i);
            const trimmedChannelData = trimmedBuffer.getChannelData(i);
            // Use subarray correctly: source.subarray(begin, end)
            trimmedChannelData.set(channelData.subarray(startOffset,
endOffset));
            }

            if (typeof toolInitializers.audioConverter.bufferToWave !== 'function') {
            showAlert('Audio conversion utility not found.', 'error');
            return;
            }
            const wavBlob =
toolInitializers.audioConverter.bufferToWave(trimmedBuffer);


            const url = URL.createObjectURL(wavBlob);
            const a = document.createElement('a');
            a.href = url;
            a.download = `${originalFileName}_trimmed.wav`;
            document.body.appendChild(a);
            a.click();
            document.body.r

emoveChild(a);
            URL.revokeObjectURL(url);
            showAlert('Audio trimmed and downloaded!', 'success');


            } catch (err) {
            showAlert(`Error trimming audio: ${err.message}`, 'error');
            console.error("Trimming error:", err);
            }
        };
        window.currentToolCleanup = () => {
            if (audioPreview && audioPreview.src &&
audioPreview.src.startsWith('blob:')) {
            URL.revokeObjectURL(audioPreview.src);
            }
        };
        },

        // ... (The rest of the toolInitializers from the previous response) ...
        // --- Age Calculator ---
```

```
ageCalculator: (container) => {
container.innerHTML = `
    <label for="birthDate">Enter your Date of Birth:</label>
    <input type="date" id="birthDate">
    <button id="calculateAgeBtn">Calculate Age</button>
    <div id="ageResult" class="result-area" style="display:none;"></div>
`;
const birthDateInput = container.querySelector('#birthDate');
const calculateBtn = container.querySelector('#calculateAgeBtn');
const ageResultDiv = container.querySelector('#ageResult');


calculateBtn.onclick = () => {
    const birthDateString = birthDateInput.value;
    if (!birthDateString) {
    showAlert('Please enter your date of birth.', 'error');
    ageResultDiv.style.display = 'none';
    return;
    }
    const birthDate = new Date(birthDateString);
    const today = new Date();


    if (birthDate > today) {
    showAlert('Birth date cannot be in the future.', 'error');
    ageResultDiv.style.display = 'none';
    return;
    }


    let years = today.getFullYear() - birthDate.getFullYear();
    let months = today.getMonth() - birthDate.getMonth();
    let days = today.getDate() - birthDate.getDate();


    if (days < 0) {
    months--;
    // Get days in the previous month of 'today'
    const prevMonth = new Date(today.getFullYear(), today.getMonth(), 0);
    days += prevMonth.getDate();
    }
    if (months < 0) {
    years--;
    months += 12;
    }
    ageResultDiv.innerHTML = `You are: <br>
            <strong>${years}</strong> years,
            <strong>${months}</strong> months, and
```

```
                    <strong>${days}</strong> days old.`;
            ageResultDiv.style.display = 'block';
            hideAlert();
        };
        },


        // --- EMI Calculator ---
        emiCalculator: (container) => {
        container.innerHTML = `
            <label for="loanAmount">Loan Amount (₹):</label>
            <input type="number" id="loanAmount" placeholder="e.g., 100000"
min="0">
            <label for="interestRate">Annual Interest R

ate (%):</label>
            <input type="number" id="interestRate" placeholder="e.g., 10.5"
min="0" step="0.01">
            <label for="loanTenure">Loan Tenure (months):</label>
            <input type="number" id="loanTenure" placeholder="e.g., 12"
min="1">
            <button id="calculateEmiBtn">Calculate EMI</button>
            <div id="emiResult" class="result-area" style="display:none;"></div>
        `;
        const amountInput = container.querySelector('#loanAmount');
        const rateInput = container.querySelector('#interestRate');
        const tenureInput = container.querySelector('#loanTenure');
        const calculateBtn = container.querySelector('#calculateEmiBtn');
        const resultDiv = container.querySelector('#emiResult');


        calculateBtn.onclick = () => {
            const P = parseFloat(amountInput.value);
            const annualRate = parseFloat(rateInput.value);
            const N = parseInt(tenureInput.value);



            if (isNaN(P) || isNaN(annualRate) || isNaN(N) || P <= 0 || annualRate <
0 || N <= 0) {
            showAlert('Please enter valid positive numbers for amount & tenure,
and a non-negative rate.', 'error');
            resultDiv.style.display = 'none';
            return;
            }

            if (annualRate === 0) { // Handle zero intere…"
```
https://yojanainformation.com/blogger-par-ai-tool-website-kaise-banaye/#:~:text=Skip%20to%20content,2025%20Yojana%20Information

I have also explained in my video how to use this given HTML script. You must watch it and in this way you will be able to create a tools converter website on your Blogger post.

Conclusion

If you want, you can also use an ad network platform and bring traffic to these tools and monetize your content with the help of an advertisement platform, which will also give you revenue.