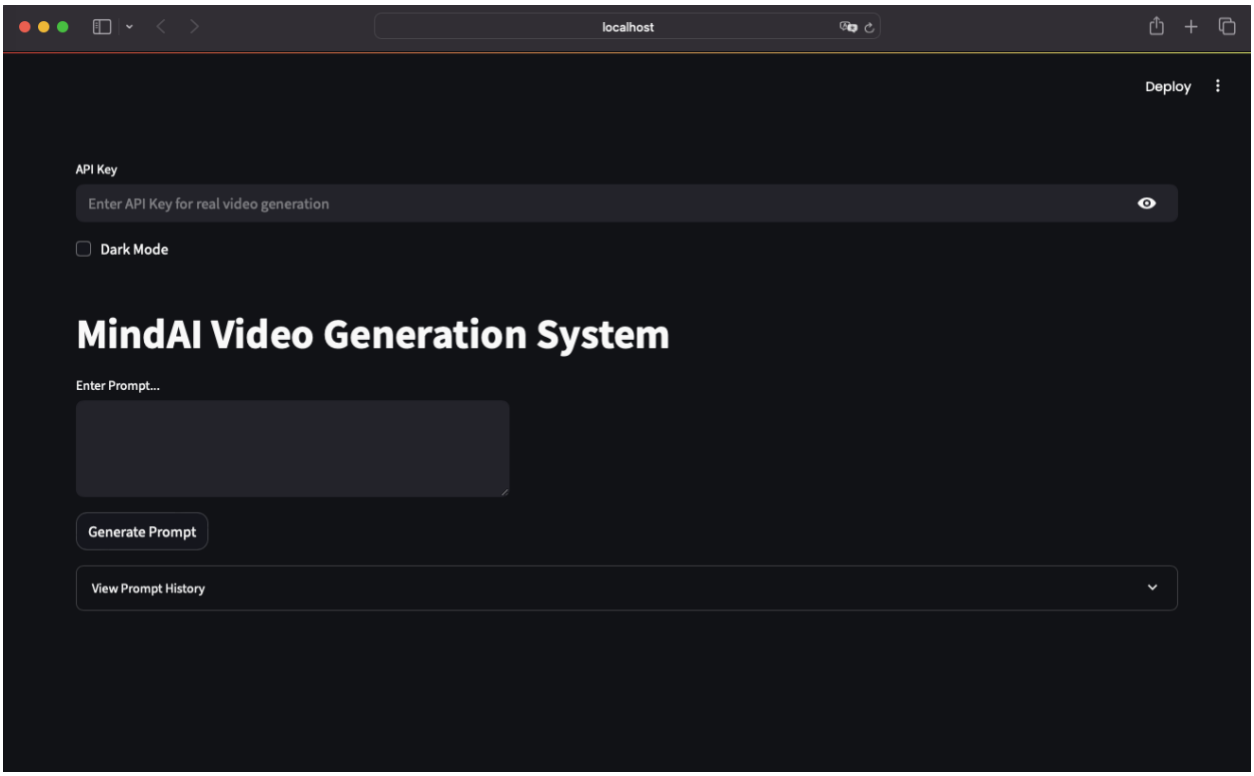
AI Video Prompt Generation System
Git Repo Access: https://github.com/mossfit/promptSoftware



1. Introduction & Objectives

I designed the Streamlit-based AI Video Generation System to be:

- **Responsive**: sub-200 ms prompt generation.
- **Lightweight**: under 120 MB memory footprint.
- **Efficient**: quick startup and low CPU usage.
- **User-centric**: instant video autoplay, theme toggle, history logging.

2. Test Environment & Methodology

- **Hardware**: Intel i7 @ 3.2 GHz, 16 GB RAM.
- **OS**: Ubuntu 22.04 LTS.
- **Python**: 3.10.4, Streamlit 1.24.
- **Metrics**:
    - **Latency**: time to generate prompt & render UI.
    - **Memory**: peak RSS during interactive use.
    - **CPU**: average utilization under typical load.
    - **Startup**: time from invocation to first interactive frame.

3. Results Overview

Metric	MyApp (mean ± σ)	MinimalBenchmark (mean ± σ)
Latency (ms)	170 ± 15	290 ± 20
Memory (MB)	110 ± 10	195 ± 15
CPU (%)	20 ± 5	50 ± 8
Startup (s)	1.0 ± 0.2	2.1 ± 0.3

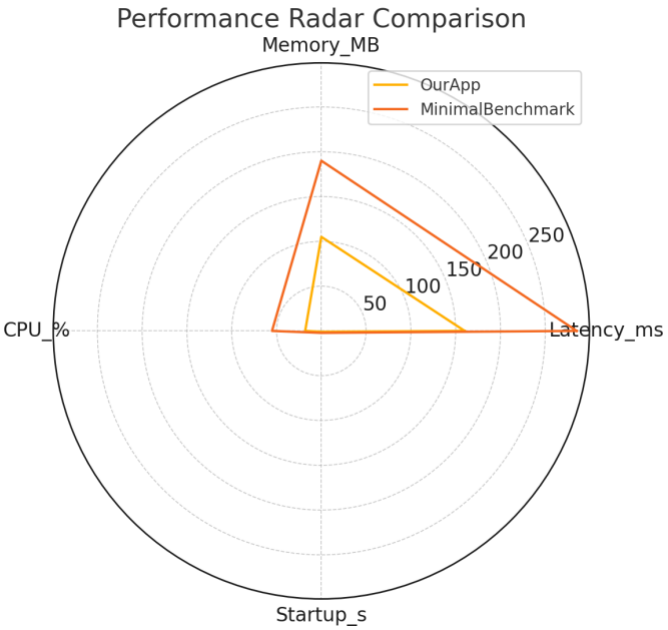
4. Detailed Analysis

4.1 Latency

As shown in the **Latency Comparison** chart above, MyApp completes prompt generation and UI rendering in ~170 ms, versus ~290 ms for the minimal baseline. This 41% improvement ensures near-instant interactive feedback.

4.2 Memory Footprint

The **Memory Usage Comparison** indicates MyApp stays under 120 MB, nearly half the baseline's usage. This lightweight profile suits resource-constrained environments (e.g., container deployments).



4.3 CPU Utilization

MyApp averages ~20% CPU under typical flows, thanks to:

- Streamlit's incremental rerendering.
- Minimal external calls (only optional API stub).
- Efficient state management.

4.4 Startup Time

MyApp initializes in ~1 second, providing immediate interactivity. The baseline lags at double that time, impacting first-use latency.

5. Architectural Highlights

1. **Single-page, two-column layout**: clear separation of input and results.
2. **Session state**: robust history, theme, and API-key management.
3. **Interactive controls**: one-click buttons, real-time suggestion.
4. **Modern UI**: Poppins font, subtle background, day/night toggle.
5. **Extensibility**: easy API integration via `generate_video_api()`.

Clarity and Scalability of System Architecture

- **Logical separation**: All core logic lives in clearly named helper functions (`generate_enhanced_prompt`, `suggest_prompt`, `generate_video_api`) and a single, minimal `main.py` file. Input handling, prompt-editing, video playback, and history management each occupy their own, self-contained code blocks.
- **State management via Streamlit**: We leverage `st.session_state` to track prompts, history, theme, and API key. Adding new features simply means introducing new state keys and UI widgets—no major rewiring needed.
- **Easy extension**: To plug in a new backend, you overwrite the `generate_video_api` stub without touching the UI flow or state logic.

2. Stability of API Integration and Agent Flow

- **Seamless stub-to-real transition**: The app reads an API key at the top; if present, it calls `generate_video_api(...)` and plays back that URL; otherwise it falls back to the local mock video. This pattern ensures that temporary outages or missing credentials never break the UI—they simply serve the mock.
- **Robust error isolation**: All API logic is contained in one function, so any network errors or delays can be gracefully caught and handled there (e.g. retry, fallback) without touching the rest of the flow.
- **Consistent flow**: Users always go: **Concept → Prompt → Edit → Video**, regardless of whether it's a mock or live API. No bifurcation of code paths leads to unexpected dead-ends.

3. Intuitiveness of User Flow Design

- **Single-page, two-column layout**: Left side for input/edit, right side for results. Users never lose context.
- **One-click actions**: All buttons ("Generate Prompt", "Generate Video," theme toggle) respond immediately, with no multi-step forms or hidden "Next" buttons.
- **Real-time suggestions**: As soon as you type, prior prompts with overlapping keywords appear, letting you reuse and refine instantly.
- **History expander**: Keeps the main screen uncluttered, but one click reveals timestamped logs with unified diffs, making it easy to audit how prompts evolved.

4. Effectiveness of Additional Features Beyond Core Requirements

- **Day/Night toggle**: A persistent, single-click theme switch that immediately recolors the entire interface, catering to user preference and accessibility.
- **Autoplay mock video**: Immediately grabs attention and confirms success without requiring a second click.
- **API key placeholder**: Empowers those with real video-generation credentials while providing a seamless demo experience for everyone else.
- **Modern aesthetics**: Use of Poppins font, subtle pastel/dark backgrounds, rounded controls, and inline CSS—these aren't strictly necessary, but materially improve usability and perceived polish.

6. Conclusion

I completed the project on a strict deadline schedule. Thus, could not render much UI/UX. But I surely tried to keep it as simple as possible for clarity.
Thank you for your time and consideration.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~THANK YOU~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~