AI Video Generation Tool (Trending News → 30–60s Video)

End-to-end prototype: News scraping → script generation → AI narration → automated video composition

Report Date: 22 Jan 2026

Deliverables: Python source code + generated artifacts (script, audio, and MP4 video) + this repor

BY- Aditya sahu

Executive Summary

This project implements a practical, one-day prototype that automatically converts a trending news article into a short video. The pipeline fetches a trending news item from an RSS feed, extracts article text, generates a short narration script using NLP summarization, converts the script into AI voice

Pipeline: News → Script → Voice → Video

architecture suitable for evaluation.

Key Outcomes

- Automated end-to-end pipeline with clear intermediate outputs (txt/mp3/mp4).
- Practical AI usage through NLP summarization (script generation) and TTS (voice narration).
- Stable and reproducible video generation using FFmpeg; output length matches narration length.
- Implementation feasible on a standard Windows laptop using free/open-source tools.

1. Problem Statement and Objectives

The assignment requires building an AI-based tool that generates short videos by scraping trending news content. The goal is to demonstrate an end-to-end pipeline that is feasible within one day, easy to understand, and produces a working video output.

Project Objectives

- Automatically fetch a trending news topic/article (data scraping).
- Generate a concise narration script suitable for a 30–60 second video (content generation).
- Generate AI narration from the script (text-to-speech).
- Create a short MP4 video by combining visuals and narration (automation).
- Keep the solution simple, practical, and reproducible.

2. Scope and Requirements

In-scope

- Trending news selection via RSS feeds.
- Text extraction for usable article content.
- Automated script generation using NLP summarization.
- AI voice narration generation using TTS.
- Video composition producing a final MP4 in ~30–60 seconds.

Out-of-scope (prototype limitations)

- Training custom ML models or using GPU-heavy generative video models.
- Complex multi-scene animations or advanced cinematic video generation.
- Full production-grade moderation, caching, and monitoring.

3. High-Level Architecture

The solution follows a modular pipeline architecture. Each module is responsible for one task and produces artifacts consumed by the next module. This improves maintainability and makes it easy to demonstrate and evaluate each stage independently.

Module	Purpose	Inputs	Outputs
fetch_news.py	Pick trending news & extract content	RSS feed URL	news_title.txt, news_text.txt
generate_script.py	Create short narration script (AI/NLP)	news_text.txt	script.txt
text_to_voice.py	Convert script to AI narration (TTS)	script.txt	voice.mp3
create_video_ffmpeg.py	Compose video from image + narration	background.jpg, voice.mp3	final_news_video.mp4

Architecture Diagram (Text)

```
[RSS Feed]
↓
(fetch_news.py) → news_title.txt, news_text.txt
↓
(generate_script.py) → script.txt
↓
(text_to_voice.py) → voice.mp3
↓
(create_video_ffmpeg.py + background.jpg) → final_news_video.mp4
```

4. Detailed Pipeline (Step-by-Step)

4.1 Step 1 – Trending News Collection (Scraping)

Goal: Automatically identify a trending news item and extract enough text to generate a short script.
Approach: Use an RSS feed to obtain trending entries (title + link) and extract full article text from the linked page.

- **Input:** RSS feed URL
- **Output:** news_title.txt (headline), news_text.txt (article content)
- **Reason:** Full text ensures the summarizer has enough context to create a meaningful script.

4.2 Step 2 – Script Generation (NLP Content Generation)

Goal: Convert long-form article text into a short narration script for a 30–60 second video.
Approach: Use extractive summarization to select key sentences and assemble them into a coherent script.

- **Input:** news_text.txt
- **Output:** script.txt
- **AI used:** automated summarization (NLP) to condense content without manual writing.

4.3 Step 3 – AI Voice Narration (Text-to-Speech)

Goal: Generate narration audio from the script.
Approach: Convert script text to speech using a TTS library and save as an MP3 file.

- **Input:** script.txt
- **Output:** voice.mp3
- **Note:** Narration duration determines final video duration.

4.4 Step 4 – Video Composition (FFmpeg Automation)

Goal: Create the final MP4 video by combining the background image and narration audio.
Approach: Use FFmpeg to loop a single image while playing narration audio; encode output as H.264 + AAC for compatibility.

- **Inputs:** background.jpg, voice.mp3
- **Output:** final_news_video.mp4
- **Robustness:** image dimensions are scaled to even values required by the H.264 encoder.

5. Setup and Running Instructions (Windows)

Prerequisites

- Python 3.10+ installed
- FFmpeg installed and available in PATH (ffmpeg -version)
- Internet access for fetching news and generating TTS audio

Install Python packages

```
python -m pip install feedparser newspaper3k lxml_html_clean sumy nltk gtts
```

Run the pipeline (in order)

```
python fetch_news.py
```

```
python generate_script.py
```

```
python text_to_voice.py
```

```
python create_video_ffmpeg.py
```

Expected final output: **final_news_video.mp4**

6. Results and Outputs

Generated artifacts

- news_title.txt – selected trending headline
- news_text.txt – extracted article text
- script.txt – narration script
- voice.mp3 – AI narration audio
- final_news_video.mp4 – final generated video

Video characteristics

- Video codec: H.264 (libx264)
- Audio codec: AAC
- Duration: matches voice.mp3 (typically 30–60 seconds)
- Playable in standard Windows media players and VLC

7. Validation and Testing

Checks performed

- Verified news_text.txt is non-empty.
- Verified script.txt contains multiple sentences.
- Verified voice.mp3 plays and is non-zero size.
- Verified final_news_video.mp4 exists and plays with audio.

Common issues and how they were handled

Issue	Cause	Mitigation
Empty article text	Some sites block scraping	Use stable RSS + extract text from trusted sources
H.264 encoder error	Odd image width/height	Auto-scale to even dimensions before encoding
FFmpeg not found	Not in PATH or shell not restarted	Validate ffmpeg -version; restart shell after install

8. Evaluation Criteria Mapping

- **Feasibility:** Lightweight pipeline; runs locally; no paid APIs required.
- **Practical AI usage:** NLP summarization for script + TTS for narration.
- **Quality & clarity:** Modular scripts and visible intermediate outputs make it easy to understand and verify.

Criterion	How the submission satisfies it
Feasibility of implementation	RSS scraping + summarization + FFmpeg composition on standard laptop
Simple, practical AI tools	Summarization (NLP) + Text-to-Speech (TTS)
Quality / ease of understanding	Clear step outputs (txt/mp3/mp4) and commands to reproduce

9. Future Enhancements

- Add subtitles automatically from script.txt (FFmpeg subtitles/drawtext).
- Use multiple images and create multi-scene videos aligned to script sentences.
- Add background music with volume ducking under narration.
- Build a simple UI (Streamlit) for one-click execution.
- Generate multiple videos for multiple trending topics in one run.

10. Conclusion

This project delivers a complete working prototype that converts trending news into a short AI-narrated video. It demonstrates a clear pipeline architecture, practical AI tools (summarization + TTS), and simple automation via FFmpeg. The outputs are easy to verify and the solution can be extended further for richer visuals.

Appendix – Project Structure and Commands

```
AI_News_Vido/
fetch_news.py
generate_script.py
text_to_voice.py
create_video_ffmpeg.py
background.jpg
news_title.txt
news_text.txt
script.txt
voice.mp3
final_news_video.mp4
```

Commands (run in order):

```
python fetch_news.py
```

```
python generate_script.py
```

```
python text_to_voice.py
```

```
python create_video_ffmpeg.py
```

References

- BBC RSS Technology feed (trending news source)
- FFmpeg documentation (video/audio composition and encoding)
- Python packages: feedparser, newspaper3k, sumy, nltk, gTTS