can you give me downloadable pdf or .md file for this document

Here is a clean, copy-paste ready Markdown document for the current working setup, including all steps to generate HD videos from Hindi/English script files inside the Ubuntu 22.04 Kubernetes pod. Save it locally as AI_Video_Agent_Runbook.md. To export as PDF, open the file in VS Code and use “Markdown: Print to PDF” or run pandoc (example: pandoc AI_Video_Agent_Runbook.md -o AI_Video_Agent_Runbook.pdf).

AI Video Agent Runbook (Kubernetes Pod, Ubuntu 22.04)

Overview
This guide documents the verified working configuration and end-to-end procedure to generate HD lip-synced videos from Hindi/English scripts using Wav2Lip inside a Kubernetes pod. It includes robust audio handling, reliable concatenation, and long-form (2–10 minute) render settings.

Key outcomes:

- Indian male voice (Hindi/English)
- Mixed scripts (auto split and process)
- Robust audio pipeline (MP3 → PCM WAV, concat demuxer)
- HD (720p) and FHD (1080p) output
- Stable Wav2Lip run for long narration

Environment

- OS: Ubuntu 22.04 (inside Kubernetes pod)
- Working directory: /code/agents/AIVideoProject
- Python: 3.11
- Core tools installed:
  - FFmpeg (with concat demuxer)
  - PyTorch CPU
  - OpenCV, librosa 0.10.1+, gTTS, soundfile, numpy (modern)
- Wav2Lip present with:
  - Wav2Lip/inference.py
  - Wav2Lip/checkpoints/wav2lip_gan.pth

    - Wav2Lip/face_detection/detection/sfd/s3fd.pth

Folder Layout (expected)

- /code/agents/AIVideoProject
    - Wav2Lip/
        - inference.py
        - checkpoints/wav2lip_gan.pth
        - face_detection/detection/sfd/s3fd.pth
    - photo.jpg
    - indian_ai_video_agent.py
    - mixed_script.txt, hindi_script.txt, indian_english_script.txt (examples)

Core Fixes Implemented

- gTTS saves audio as MP3 by default; pipeline now explicitly converts to true PCM WAV mono 24 kHz before concatenation.
- Audio concatenation uses FFmpeg concat demuxer with a file list to avoid filter_complex pitfalls and “-i” placement errors.
- Modern librosa (0.10.1+) used with current numpy; hparams completed so mel filter invocation works with Wav2Lip.
- Long-form settings for Wav2Lip: pads and nosmooth tuned for multi-minute scripts.

One-time Verification

- Ensure model files exist:
    - ls -la Wav2Lip/inference.py
    - ls -la Wav2Lip/checkpoints/wav2lip_gan.pth
    - ls -la Wav2Lip/face_detection/detection/sfd/s3fd.pth
- Check FFmpeg:
    - ffmpeg -version
- Check Python libs:
    - python3 -c "import torch, cv2, librosa, soundfile, gtts; print('OK')"

Script File Preparation

- Use UTF-8 encoding for Hindi text.
- Keep sentences concise with punctuation (. ! ? ।).
- Mixed scripts may interleave Hindi (Devanagari) and English sentences.

Examples:
mixed_script.txt
Namaste! मैं आपका स्वागत करता हूँ। Welcome to my channel where I discuss technology और business के बारे में।

Today हम बात करेंगे artificial intelligence की। AI has revolutionized हमारी जिंदगी को। From smartphones से लेकर smart homes तक, everything is connected.

भविष्य में, AI will help us solve major problems। Climate change, healthcare, और education - सभी क्षेत्रों में AI का योगदान होगा।

Thank you for watching! अगर यह video पसंद आया, तो please like और subscribe करें। Dhanyawad!

hindi_script.txt
नमस्कार दोस्तों! आज हम तकनीक की महत्वपूर्ण भूमिका पर चर्चा करेंगे।
स्मार्टफोन, इंटरनेट और एआई ने हमारे जीवन को आसान बना दिया है।
शिक्षा और स्वास्थ्य क्षेत्र में भी एआई का योगदान बढ़ रहा है।
धन्यवाद!

indian_english_script.txt
Hello everyone! Today’s topic is the rise of AI in India.
From fintech to healthtech, startups are innovating across the country.
With Digital India, opportunities are expanding for young entrepreneurs.
Thank you for watching!

Final Agent (drop-in)
Place this file as indian_ai_video_agent.py in /code/agents/AIVideoProject.

```
#!/usr/bin/env python3
import os
import subprocess
import sys
from pathlib import Path
import shutil
import re

def sh(cmd: list, check=True, capture=False):
return subprocess.run(cmd, check=check, capture_output=capture, text=True)

def check_wav2lip_setup():
required = [
"Wav2Lip/inference.py",
"Wav2Lip/checkpoints/wav2lip_gan.pth",
"Wav2Lip/face_detection/detection/sfd/s3fd.pth",
]
print("🔍 Checking Wav2Lip setup...")
missing = []
for p in required:
if Path(p).exists():
sz = Path(p).stat().st_size / (1024 * 1024)
print(f"✅ {p} ({sz:.1f}MB)")
else:
print(f"❌ Missing: {p}")
```

```
missing.append(p)
return len(missing) == 0

def read_script_file(script_path):
script_path = Path(script_path)
if not script_path.exists():
print(f"✗ Script file not found: {script_path}")
return None
encodings = ["utf-8", "utf-16", "iso-8859-1"]
for enc in encodings:
try:
content = script_path.read_text(encoding=enc).strip()
if content:
print(f"✓ Script loaded ({len(content)} chars) with {enc} encoding")
return content
except UnicodeDecodeError:
continue
print("✗ Failed to read script with any encoding")
return None

def detect_script_language(text: str):
hindi_pat = re.compile(r"[\u0900-\u097F]")
english_pat = re.compile(r"[a-zA-Z]")
has_hi = bool(hindi_pat.search(text))
has_en = bool(english_pat.search(text))
if has_hi and has_en:
return "mixed"
if has_hi:
return "hindi"
if has_en:
return "english"
return "english"

def split_mixed_script(text: str):
sentences = re.split(r"[.!?।]", text)
segments = []
for s in sentences:
s = s.strip()
if not s:
continue
lang = detect_script_language(s)
segments.append({
"text": s,
"language": "hi" if lang == "hindi" else "en",
"tld": "co.in" if lang in ["english", "mixed"] else None
})
print(f" Script split into {len(segments)} segments")
```

```
for i, seg in enumerate(segments[:3]):
lang_name = "Hindi" if seg["language"] == "hi" else "English"
print(f" {i+1}. {lang_name}: {seg['text'][:50]}...")
return segments

def generate_indian_male_tts(text, out_wav_path, language="en", tld="co.in"):
try:
from gtts import gTTS
tmp_mp3 = str(out_wav_path) + ".mp3"
if language == "hi":
tts = gTTS(text=text, lang="hi", slow=False)
else:
tts = gTTS(text=text, lang="en", tld=tld or "co.in", slow=False)
tts.save(tmp_mp3)
sh(["ffmpeg", "-y", "-i", tmp_mp3, "-ac", "1", "-ar", "24000", "-c:a", "pcm_s16le",
str(out_wav_path)])
Path(tmp_mp3).unlink(missing_ok=True)
if Path(out_wav_path).exists():
sz = Path(out_wav_path).stat().st_size
print(f"✅ Audio saved: {out_wav_path} ({sz} bytes)")
return True
print(f"❌ Failed to create {out_wav_path}")
return False
except Exception as e:
print(f"❌ TTS generation failed: {e}")
return False

def process_mixed_script_audio(segments, base_name="segment"):
files = []
for i, seg in enumerate(segments):
f = f"{base_name}_{i:03d}.wav"
ok = generate_indian_male_tts(seg["text"], f, seg["language"], seg.get("tld"))
if ok:
files.append(f)
else:
print(f"⚠️ Skipping failed segment {i}")
return files

def concatenate_audio_files(audio_files, output_path, add_pauses=True):
try:
if not audio_files:
print("❌ No audio files to concatenate")
return False
norm_files = []
for f in audio_files:
nf = f"norm_{Path(f).stem}.wav"
sh(["ffmpeg", "-y", "-i", f, "-ac", "1", "-ar", "24000", "-c:a", "pcm_s16le", nf])
```

```
norm_files.append(nf)
if add_pauses:
if not Path("silence.wav").exists():
sh(["ffmpeg", "-y", "-f", "lavfi", "-t", "0.5", "-i", "anullsrc=r=24000:cl=mono", "-c:a",
"pcm_s16le", "silence.wav"])
with open("audio_list.txt", "w") as f:
for i, nf in enumerate(norm_files):
f.write(f"file '{nf}'\n")
if add_pauses and i < len(norm_files) - 1:
f.write("file 'silence.wav'\n")
sh(["ffmpeg", "-y", "-f", "concat", "-safe", "0", "-i", "audio_list.txt", "-c:a", "pcm_s16le",
str(output_path)])
Path("audio_list.txt").unlink(missing_ok=True)
for nf in norm_files:
Path(nf).unlink(missing_ok=True)
dur = 0.0
r = sh(["ffprobe", "-v", "quiet", "-show_entries", "format=duration", "-of", "csv=p=0",
str(output_path)], check=False, capture=True)
if r.returncode == 0 and r.stdout.strip():
dur = float(r.stdout.strip())
print(f"✅ Audio concatenated: {output_path} (Duration: {dur:.1f}s)")
return True
except subprocess.CalledProcessError as e:
print(f"❌ Audio concatenation failed: {e.stderr or e}")
return False
except Exception as e:
print(f"❌ Error concatenating audio: {e}")
return False

def enhance_indian_male_voice(input_audio, output_audio):
try:
af = (
"volume=1.3,"
"highpass=f=85,"
"lowpass=f=7500,"
"equalizer=f=200:width_type=h:width=100:g=2,"
"equalizer=f=1000:width_type=h:width=200:g=1,"
"equalizer=f=3000:width_type=h:width=500:g=-0.5,"
"compand=0.1:0.2:-50/-10|-20/-5:0.05:0:-90:0.1"
)
r = sh(["ffmpeg", "-y", "-i", str(input_audio), "-af", af, str(output_audio)], check=False,
capture=True)
if r.returncode == 0:
print(f"✅ Voice enhanced: {output_audio}")
return True
print(f"⚠️ Enhancement failed, copying original")
```

```
shutil.copy2(input_audio, output_audio)
return True
except Exception as e:
print(f"✗ Voice enhancement failed: {e}")
return False

def run_wav2lip_hd(image_path, audio_path, output_path):
try:
abs_img = Path(image_path).resolve()
abs_aud = Path(audio_path).resolve()
abs_out = Path(output_path).resolve()
if not abs_img.exists():
print(f"✗ Image file not found: {abs_img}")
return False
if not abs_aud.exists():
print(f"✗ Audio file not found: {abs_aud}")
return False
dur = 0.0
r = sh(["ffprobe", "-v", "quiet", "-show_entries", "format=duration", "-of", "csv=p=0",
str(abs_aud)], check=False, capture=True)
if r.returncode == 0 and r.stdout.strip():
dur = float(r.stdout.strip())
print("🎬 Running Wav2Lip HD processing...")
print(f"📸 Image: {abs_img}")
print(f"🎵 Audio: {abs_aud} (Duration: {dur:.1f}s)")
print(f"📁 Output: {abs_out}")
cmd = [
"python3", "inference.py",
"--checkpoint_path", "checkpoints/wav2lip_gan.pth",
"--face", str(abs_img),
"--audio", str(abs_aud),
"--outfile", str(abs_out),
"--pads", "0", "15", "0", "0",
"--resize_factor", "1",
"--nosmooth",
]
r2 = subprocess.run(cmd, cwd="Wav2Lip", capture_output=True, text=True)
if r2.returncode == 0:
if abs_out.exists():
sz = abs_out.stat().st_size / (1024 * 1024)
print(f"✅ HD Video generated: {abs_out} ({sz:.1f}MB)")
return True
print("✗ Output file not created despite success code")
return False
print(f"✗ Wav2Lip failed\nSTDOUT:\n{r2.stdout}\nSTDERR:\n{r2.stderr}")
return False
```

except Exception as e:
print(f"✗ Error running Wav2Lip: {e}")
return False

def upscale_video_to_hd(input_video, output_video, target="hd"):
try:
scale = "1280:720" if target == "hd" else "1920:1080"
r = sh([
"ffmpeg", "-y", "-i", str(input_video),
"-vf", f"scale={scale}:flags=lanczos",
"-c:v", "libx264", "-preset", "slow", "-crf", "18",
"-c:a", "aac", "-b:a", "128k",
str(output_video)
], check=False, capture=True)
if r.returncode == 0:
sz = Path(output_video).stat().st_size / (1024 * 1024)
print(f"✓ {target. upper()} video created: {output_video} ({sz:.1f}MB)")
return True
print(f"✗ Upscaling failed:\n{r.stderr}")
return False
except Exception as e:
print(f"✗ Error upscaling video: {e}")
return False

def main():
import argparse
parser = argparse.ArgumentParser(description="Indian AI Video Agent - Hindi/English HD Videos (concat fixed)")
parser.add_argument("--script-file", required=True, help="Path to script file (.txt)")
parser.add_argument("--image", required=True, help="Path to image file")
parser.add_argument("--output", default="indian_hd_video.mp4", help="Output video path")
parser.add_argument("--language", choices=["auto", "hindi", "english", "mixed"], default="auto", help="Script language")
parser.add_argument("--quality", choices=["hd", "fhd"], default="hd", help="Output quality (hd=720p, fhd=1080p)")
parser.add_argument("--add-pauses", action="store_true", help="Add pauses between segments")
args = parser.parse_args()

```
print("🇮🇳 Indian AI Video Agent - Hindi/English HD Videos (concat fixed)")
print(f"📁 Working in: {Path.cwd()}")
print("=" * 70)

if not check_wav2lip_setup():
    return

if not Path(args.image).exists():
    print(f"✗ Image not found: {Path(args.image).resolve()}")
```

```
    return

content = read_script_file(args.script_file)
if not content:
    return

detected = detect_script_language(content)
language = args.language if args.language != "auto" else detected
print(f"  Detected language: {detected}")
print(f" Processing as: {language}")

base_audio = Path("indian_base_audio.wav")
enhanced_audio = Path("indian_enhanced_audio.wav")
raw_video = Path("raw_output.mp4")
for p in [base_audio, enhanced_audio, raw_video]:
    p.unlink(missing_ok=True)

try:
    if language == "mixed":
        print(" Processing mixed Hindi-English script...")
        segments = split_mixed_script(content)
        seg_files = process_mixed_script_audio(segments)
        if not seg_files:
            print("✗ No audio segments generated")
            return
        if not concatenate_audio_files(seg_files, base_audio, args.add_pauses):
            return
        for f in seg_files:
            Path(f).unlink(missing_ok=True)
    else:
        lang_code = "hi" if language == "hindi" else "en"
        tld = "co.in" if language == "english" else None
        if not generate_indian_male_tts(content, base_audio, lang_code, tld):
            return

    if not enhance_indian_male_voice(base_audio, enhanced_audio):
        return

    print("\n Starting HD video generation...")
    if not run_wav2lip_hd(args.image, enhanced_audio, raw_video):
        return

    if args.quality == "fhd":
        if not upscale_video_to_hd(raw_video, args.output, target="fhd"):
            return
    else:
        r = subprocess.run([
            "ffmpeg", "-y", "-i", str(raw_video),
            "-c:v", "libx264", "-preset", "slow", "-crf", "20",
            "-c:a", "aac", "-b:a", "128k",
            str(args.output)
        ], capture_output=True, text=True)
        if r.returncode != 0:
            print("⚠  Final encoding failed, using raw output")
            shutil.copy2(raw_video, args.output)
```

```
    final = Path(args.output).resolve()
    if final.exists():
        sz = final.stat().st_size / (1024 * 1024)
        info = subprocess.run([
            "ffprobe", "-v", "quiet",
            "-show_entries", "format=duration,size:stream=width,height",
            "-of", "csv=p=0", str(final)
        ], capture_output=True, text=True)
        print("\n SUCCESS! Indian HD video created!")
        print("=" * 50)
        print(f" Location: {final}")
        print(f" Size: {sz:.1f}MB")
        if info.returncode == 0:
            lines = info.stdout.strip().split("\n")
            if len(lines) >= 2:
                try:
                    duration = float(lines.split(","))
                    width, height = lines.split(",")[0:2][^21]
                    print(f"  Duration: {duration:.1f} seconds ({duration/60:.1f} minute
                    print(f" Resolution: {width}x{height}")
                except Exception:
                    pass
        print("")
        print(" Voice Features Applied:")
        print(f"   Language: {language.title()}")
        print("   Voice: Indian Male")
        print("   Enhancement: Professional audio processing")
        print("   Quality:", "HD (720p)" if args.quality == "hd" else "Full HD (1080p)")
except Exception as e:
    print(f"✖ Unexpected error: {e}")
finally:
    print("\n Cleaning up temporary files...")
    for p in [base_audio, enhanced_audio, raw_video]:
        p.unlink(missing_ok=True)
```

if **name** == "**main**":
main()

Usage: Generate Videos

Mixed Hindi-English (HD, with pauses)
python3 indian_ai_video_agent.py
--script-file mixed_script.txt
--image photo.jpg
--language mixed
--quality hd
--add-pauses
--output mixed_hindi_english_video.mp4

Hindi (FHD)
python3 indian_ai_video_agent.py
--script-file hindi_script.txt
--image photo.jpg

--language hindi
--quality fhd
--output hindi_fhd_video.mp4

Indian English (HD)
python3 indian_ai_video_agent.py
--script-file indian_english_script.txt
--image photo.jpg
--language english
--quality hd
--output indian_english_hd.mp4

Copying files in/out of the pod

- Copy image or script into pod:
  - kubectl cp local_photo.jpg <pod-name>:/code/agents/AIVideoProject/photo.jpg
  - kubectl cp mixed_script.txt <pod-name>:/code/agents/AIVideoProject/mixed_script.txt
- Copy final video out:
  - kubectl cp <pod-name>:/code/agents/AIVideoProject/mixed_hindi_english_video.mp4 ./mixed_hindi_english_video.mp4

Troubleshooting

- No such file or directory during audio join
  - Cause: Wrong filter_complex or orphan “-i”. Fix: Use concat demuxer with a properly formatted file list (already implemented).
- Audio concatenation produces glitches
  - Ensure all segments are PCM s16le mono 24 kHz before concatenation (handled by normalization in the script).
- Hindi characters not read correctly
  - Save script as UTF-8. The agent tries utf-8, utf-16, then iso-8859-1; UTF-8 is recommended.
- Long video performance
  - HD (720p) is faster; FHD (1080p) increases time and file size. For 2–10 minutes, expect multiple minutes of processing.

Exporting to PDF

- VS Code: Open AI_Video_Agent_Runbook.md, run “Markdown: Print to PDF”.
- pandoc: pandoc AI_Video_Agent_Runbook.md -o AI_Video_Agent_Runbook.pdf

End of document

⁂

1. https://www.semanticscholar.org/paper/c3fae080029a6fb8a8ffd445043c94139e3e0c25

2. https://www.semanticscholar.org/paper/ea51ef5810e003ba64875ec3f04d8892c3f3da48
3. http://arxiv.org/pdf/2406.08096.pdf
4. http://arxiv.org/pdf/2110.11499v2.pdf
5. https://arxiv.org/pdf/2110.08580.pdf
6. http://arxiv.org/pdf/2305.12328.pdf
7. http://arxiv.org/pdf/2401.04468.pdf
8. http://arxiv.org/pdf/2405.20412.pdf
9. http://arxiv.org/pdf/2403.14773.pdf
10. https://arxiv.org/pdf/2305.00521.pdf
11. http://arxiv.org/pdf/2201.03809.pdf
12. https://www.semanticscholar.org/paper/3f5d93b33d8b33a4b67543d5bd2ce88ab0c45f48
13. https://arxiv.org/pdf/1803.10404.pdf
14. https://www.semanticscholar.org/paper/6aaa198837c2325db31c0513a0abbf36013dde35
15. https://www.semanticscholar.org/paper/dd5051912349fba0a7cefd328f8bf1f7074f847c
16. https://www.semanticscholar.org/paper/55a3fcb76fe8a3a0b0f1a0005daefd39165fc297
17. https://www.semanticscholar.org/paper/f320316f4f60dc208a75d3296e8db80989d86a06
18. https://www.tandfonline.com/doi/full/10.1080/07391102.2023.2239929
19. https://journals.lww.com/10.4103/apc.apc_38_23
20. https://www.semanticscholar.org/paper/457f1d831f53aadc124b226b3ee29e6a8deb12ca