🚀 Project: Science News Shorts Automation (AI + Kotlin + Python MSA)

File Type: Vibe Coding Master Manifest

Target: Anti-Gravity IDE / Context

Description: 비용 0원으로 과학 기술 뉴스를 수집하고, AI(Stable Video Diffusion)로 영상을 생성하여 유튜브 숏츠를 자동 제작하는 MSA 시스템 전체 명세입니다.

📂 1. Project Architecture & Strategy

1.1 Tech Stack (Zero Cost)

- **Controller:** Kotlin (JVM) - 비즈니스 로직, 오케스트레이션
- **AI Media Service:** Python (FastAPI) - GPU 가속 AI 처리
  - **Video (I2V):** Stable Video Diffusion (SVD-XT) - 이미지 → 4초 동영상 변환
  - **Audio (TTS):** Edge-TTS - 고품질 한국어 음성 생성
- **Infrastructure:** Docker & Docker Compose (GPU Pass-through)

1.2 Workflow

1. **News Fetch:** Kotlin이 RSS 피드에서 최신 과학 뉴스를 수집.
2. **Scripting:** Gemini API(Flash)를 이용해 3초 훅 + 40초 본문 대본 작성.
3. **Visuals:** 키워드로 Pexels 등에서 고화질 이미지 다운로드.
4. **AI Generation:**
   - Kotlin → Python: 이미지 전송 → SVD 모델이 "움직이는 비디오" 생성.
   - Kotlin → Python: 텍스트 전송 → TTS 오디오 생성.
5. **Editing:** FFmpeg로 AI 비디오 클립 연결 + 오디오 합성.
6. **Upload:** YouTube Data API로 업로드.

1.3 News Source Strategy

저작권 이슈를 최소화하고 지속적인 콘텐츠 수급을 위한 뉴스 소스 선정 전략입니다.

🌍 해외 소스 **(추천: 콘텐츠 품질 및 수량 우수)**

사이트명	특징	**RSS URL (예시)**	비고
ScienceDaily	**(강력 추천)** 과학 전 분야 요약 우수	https://www.sciencedaily.com/rss/top_news.xml	주제별 세분화 가능

NASA	우주/항공, 퍼블릭 도메인 이미지	https://www.nasa.gov/rss/dyn/breaking_news.rss	저작권 걱정 없음
TechCrunch	최신 IT/스타트업 트렌드	https://techcrunch.com/feed/	신기술 비즈니스 위주
Wired	대중적이고 흥미로운 제목(Hook)	https://www.wired.com/feed/category/science/latest/rss	일반 대중 타겟팅 유리

🇰🇷 국내 소스

사이트명	특징	**RSS URL (예시)**	비고
NST (국가과학기술연구회)	정부출연연구기관 성과 모음	https://www.nst.re.kr/nst/rss/rss.jsp	공신력 높음
Google News	특정 키워드 기반 뉴스 수집	https://news.google.com/rss/search?q={키워드}	선별 과정 필요

⚠️ 기술적 고려사항 **(Dev Tips)**

1. **Partial Feed** 대응**:** 대부분의 RSS는 요약만 제공하므로, Jsoup 라이브러리로 원문 링크(link)에 접속하여 본문(<body>)을 크롤링하는 로직이 필요함.
2. 저작권 안전장치**:**
   - 기사 원문을 그대로 읽지 말고, 반드시 **Gemini**에게 **"**완전히 새로운 문장으로 요약 및 재구성**(Paraphrasing)"** 하도록 프롬프트 작성.
   - 영상 설명란에 원문 출처(Source Link) 자동 기재.

💻 2. Application Code (Kotlin Controller)

kotlin-controller/src/main/kotlin/ShortsMaker.kt

> 메인 로직: 뉴스 수집부터 업로드까지의 흐름을 제어합니다. Python 서비스와 HTTP로 통신합니다.

```
import java.io.File
import java.util.concurrent.TimeUnit
import okhttp3.*
import okhttp3.MediaType.Companion.toMediaType
```

```
import okhttp3.RequestBody.Companion.asRequestBody
import org.json.JSONObject

// --- Data Models ---
data class NewsData(val title: String, val link: String, val summary: String)
data class ScriptData(val hook: String, val body: String, val keywords: List<String>)

// --- Service Interfaces ---
interface MediaService {
    fun generateAudio(text: String, outputFile: File): File
    fun generateVideo(imageFile: File, outputFile: File): File
}

// --- Implementation (Communicating with Python MSA) ---
class PythonMediaService(private val serviceUrl: String) : MediaService {
    private val client = OkHttpClient.Builder()
        .connectTimeout(5, TimeUnit.MINUTES) // SVD 생성 시간 고려
        .readTimeout(5, TimeUnit.MINUTES)
        .build()

    override fun generateAudio(text: String, outputFile: File): File {
        val json = JSONObject().put("text", text).put("voice", "ko-KR-SunHiNeural").toString()
        val request = Request.Builder()
            .url("$serviceUrl/generate-audio")
            .post(RequestBody.create("application/json".toMediaType(), json))
            .build()

        client.newCall(request).execute().use { response ->
            if (!response.isSuccessful) throw RuntimeException("TTS Failed: ${response.code}")
            // 실제 구현: 응답으로 받은 파일 URL 다운로드 또는 바이너리 저장
            // 여기서는 Mocking: 파일이 생성되었다고 가정
            println("Audio generated via Python Service")
        }
        return outputFile
    }

    override fun generateVideo(imageFile: File, outputFile: File): File {
        val requestBody = MultipartBody.Builder()
            .setType(MultipartBody.FORM)
            .addFormDataPart("file", imageFile.name,
imageFile.asRequestBody("image/jpeg".toMediaType()))
            .build()
```

```
        val request = Request.Builder()
            .url("$serviceUrl/generate-video")
            .post(requestBody)
            .build()

        println("Requesting AI Video Generation (This may take time)...")
        client.newCall(request).execute().use { response ->
            if (!response.isSuccessful) throw RuntimeException("SVD Video Failed:
${response.code}")
            // 실제 구현: 응답 JSON에서 filename 파싱 후 다운로드
            println("Video generated via Python Service (SVD)")
        }
        return outputFile
    }
}

// --- Main Orchestrator ---
fun main() {
    val workspace = File("workspace").apply { mkdirs() }
    val pythonServiceUrl = "http://localhost:8000" // Docker Compose 서비스명 사용 시 수정
필요
    val mediaService = PythonMediaService(pythonServiceUrl)

    try {
        println("=== Shorts Automation Start ===")

        // 1. Mock Data (News & Script)
        val script = ScriptData(
            "AI가 영상을 만듭니다!",
            "이 영상은 정지된 이미지를 AI가 움직이게 만든 것입니다.",
            listOf("robot", "future")
        )

        // 2. Prepare Resources
        val imageFile = File(workspace, "input_image.jpg") // 미리 준비된 이미지
        if (!imageFile.exists()) {
            println("Place an 'input_image.jpg' in workspace folder to test.")
            return
        }

        // 3. AI Generation
        val audioFile = mediaService.generateAudio(script.body, File(workspace, "audio.mp3"))
        val videoFile = mediaService.generateVideo(imageFile, File(workspace, "ai_video.mp4"))
```

```
        println("=== Process Complete ===")
        println("Video: ${videoFile.absolutePath}")
        println("Audio: ${audioFile.absolutePath}")

    } catch (e: Exception) {
        e.printStackTrace()
    }
}
```

🐍 3. AI Media Service (Python Microservice)

ai_media_service/app.py

FastAPI 서버: Edge-TTS와 Stable Video Diffusion(SVD) 모델을 구동합니다.

```
from fastapi import FastAPI, HTTPException, UploadFile, File
from pydantic import BaseModel
import edge_tts
import torch
from diffusers import StableVideoDiffusionPipeline
from diffusers.utils import load_image, export_to_video
import os
import uuid
import uvicorn
from PIL import Image
import io

app = FastAPI(title="Shorts AI Media Service")

# --- Config ---
OUTPUT_DIR = "/app/output"
os.makedirs(OUTPUT_DIR, exist_ok=True)

# --- Load AI Model (SVD) ---
print("Loading SVD Model... This might take a while.")
try:
    pipe = StableVideoDiffusionPipeline.from_pretrained(
        "stabilityai/stable-video-diffusion-img2vid-xt",
        torch_dtype=torch.float16,
        variant="fp16"
    )
```

```
    if torch.cuda.is_available():
        pipe.enable_model_cpu_offload() # VRAM 최적화
        print("✅ SVD Model loaded on CUDA.")
    else:
        print("⚠️ WARNING: CUDA not found. SVD will be extremely slow.")
except Exception as e:
    print(f"❌ Failed to load SVD model: {e}")
    pipe = None

# --- Models ---
class TTSRequest(BaseModel):
    text: str
    voice: str = "ko-KR-SunHiNeural"

# --- Endpoints ---
@app.get("/")
def health_check():
    gpu_status = "available" if torch.cuda.is_available() else "unavailable"
    return {"status": "ok", "gpu": gpu_status}

@app.post("/generate-audio")
async def generate_audio(request: TTSRequest):
    try:
        filename = f"audio_{uuid.uuid4()}.mp3"
        output_path = os.path.join(OUTPUT_DIR, filename)
        communicate = edge_tts.Communicate(request.text, request.voice)
        await communicate.save(output_path)
        return {"status": "success", "filename": filename}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

@app.post("/generate-video")
async def generate_video(file: UploadFile = File(...)):
    if pipe is None:
        raise HTTPException(status_code=503, detail="Video model not loaded")

    try:
        # 1. Read & Resize Image
        contents = await file.read()
        image = Image.open(io.BytesIO(contents)).convert("RGB")
        image = image.resize((576, 1024)) # Shorts Ratio Target

        # 2. Generate Video (Inference)
```

```
        # motion_bucket_id controls the amount of motion (default 127)
        frames = pipe(image, decode_chunk_size=8, generator=torch.manual_seed(42),
motion_bucket_id=127).frames[0]

        # 3. Save Video
        filename = f"video_{uuid.uuid4()}.mp4"
        output_path = os.path.join(OUTPUT_DIR, filename)
        export_to_video(frames, output_path, fps=7)

        return {"status": "success", "filename": filename}

    except Exception as e:
        print(f"Video Generation Error: {e}")
        raise HTTPException(status_code=500, detail=str(e))

if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

🐳 4. Infrastructure Configuration

ai_media_service/Dockerfile

```
        GPU 지원을 위한 PyTorch 베이스 이미지

FROM pytorch/pytorch:2.1.0-cuda12.1-cudnn8-runtime

WORKDIR /app

# System dependencies
RUN apt-get update && apt-get install -y \
    libgl1-mesa-glx \
    libglib2.0-0 \
    && rm -rf /var/lib/apt/lists/*

# Python dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# App Source
COPY app.py .

# Output Volume
```

```
RUN mkdir -p /app/output

EXPOSE 8000
CMD ["python", "app.py"]
```

ai_media_service/requirements.txt

```
fastapi
uvicorn
pydantic
python-multipart
edge-tts
torch
diffusers
transformers
accelerate
opencv-python-headless
pillow
```

docker-compose.yml

GPU 리소스 할당 및 서비스 연결

```
services:
  ai-media-service:
    build: ./ai-media-service
    container_name: shorts-ai-service
    ports:
      - "8000:8000"
    volumes:
      - ./shared-data:/app/output
      - ~/.cache/huggingface:/root/.cache/huggingface # Model Caching
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]
    restart: unless-stopped

  # Kotlin 앱은 로컬 실행 또는 별도 컨테이너 추가 가능
```

📝 5. Execution Guide

1. 사전 준비:
   - Docker Desktop 설치 & 실행
   - NVIDIA GPU 드라이버 및 nvidia-container-toolkit 설치 확인
2. 실행:
   docker-compose up --build

3. 테스트:
   - workspace 폴더에 테스트용 input_image.jpg 넣기.
   - Kotlin 프로젝트 실행 (ShortsMaker.kt의 main 함수).
   - shared-data 폴더(Docker 볼륨)에 생성된 .mp4 확인.