# AI CONTENT REPURPOSING AGENT

## Project Documentation

---

**Student Name:** Sai Prasad
**Project:** AI Agent for Automated Content Repurposing
**Date:** October 19, 2025
**Assignment:** Task 5 - AI Agent Challenge

---

## 1. EXECUTIVE SUMMARY

### Problem

Content creators spend 2-3 hours manually adapting content for different platforms (LinkedIn, Twitter, YouTube). Each platform needs different formats, lengths, and tones.

### Solution

An AI-powered agent that automatically transforms content from URLs, PDFs, YouTube videos, or text files into 4 platform-optimized formats using Google's Gemini AI.

### Output Formats

- LinkedIn posts (with hooks and CTAs)
- Twitter threads (5 tweets)
- YouTube scripts (2-3 minutes)
- Summaries with key insights

### Impact

- **95% time savings** (2 hours → 5 minutes)
- Consistent quality across platforms
- Scalable for content teams

---

## 2. TOOLS & LIBRARIES USED

### Core Technologies

| Technology | Purpose |
|---|---|
| Python 3.13 | Programming language |
| Google Gemini API | AI content generation |
| google-genai SDK | API integration |

### Content Extraction

| Library | Purpose |
|---|---|
| requests | HTTP requests for web pages |
| BeautifulSoup4 | HTML parsing |
| pdfplumber | PDF text extraction |
| youtube-transcript-api | YouTube transcripts |

### Supporting Tools

| Library | Purpose |
|---|---|
| python-dotenv | Environment variables |
| time | Rate limit management |
| argparse | Command-line interface |

---

# 3. STEP-BY-STEP WORKFLOW

## Phase 1: Content Extraction

**Input Methods:**

```
python agent.py --url "https://example.com/article"
python agent.py --pdf "document.pdf"
python simple_agent.py --file "article.txt"
```

**Process:**

1. User provides content source via command line
2. System routes to appropriate extractor:
    - URLs → Web scraping with BeautifulSoup

    - PDFs → Text extraction with pdfplumber
    - Text files → Direct file reading
3. Content is cleaned (remove scripts, styles, extra whitespace)
4. Output: Clean, formatted text

## Phase 2: Content Processing

**Smart Optimization:**

- Content < 3,000 chars → Use directly
- Content > 3,000 chars → Trim to first 3,000 characters

**Why?**

- API has token limits
- Free tier: 50 requests/day
- Shorter content = fewer API calls = stays within quota

## Phase 3: AI Generation

**4 API Calls with Rate Limiting:**

1. **LinkedIn Post** (wait 6 seconds)
    - Professional tone
    - Hook + 2 insights + CTA
    - Max 200 words
2. **Tweet Thread** (wait 6 seconds)
    - 5 numbered tweets (1/5 to 5/5)
    - Each < 280 characters
    - Conversational tone
3. **YouTube Script** (wait 6 seconds)
    - 10s hook + 3 main points + CTA
    - 2-3 minute duration
    - Conversational style
4. **Summary**
    - Key insights (bullet points)
    - Main takeaway
    - Target audience

**Rate Limiting:**

```
generate_linkedin()
time.sleep(6)  # 10 requests/min = 6s between calls
generate_tweets()
time.sleep(6)
```

# ... continues

### Phase 4: Output

**Display Format:**

```
============================================================
✅ REPURPOSING COMPLETE!
============================================================

📌 LINKEDIN
[LinkedIn post content]

📌 TWEETS
[Tweet thread content]

📌 YOUTUBE
[YouTube script content]

📌 SUMMARY
[Summary content]
```

---

## 4. HOW THE AI AGENT WORKS

### Architecture

```
User Input (CLI)
      ↓
Content Extractor (URLs/PDFs/Files)
      ↓
Text Processing (Clean & Optimize)
      ↓
Gemini API (4 Generation Calls)
      ↓
Format & Display Results
```

### File Structure

```
Task 5/
├── agent.py              # Main orchestrator
├── simple_agent.py         # Simplified for demo
├── gemini_client.py        # API wrapper
├── content_extractors.py    # Extraction functions
├── prompts.py             # Prompt templates
```

```
├── test.py              # Connection test
├── requirements.txt     # Dependencies
├── .env                 # API keys (not committed)
├── sample_article.txt   # Demo content
└── README.md            # Instructions
```

### Key Functions

**gemini_client.py:**

- `make_client()` - Authenticates with API key
- `generate_text()` - Sends prompts, returns AI responses

**content_extractors.py:**

- `extract_text_from_url()` - Web scraping
- `extract_text_from_pdf()` - PDF parsing
- `extract_from_youtube()` - Transcript fetching

**prompts.py:**

- Contains template prompts for each format
- Clear structure + specific requirements

### Prompt Engineering

Each prompt includes:

1. **Role:** "You are an expert writer..."
2. **Task:** "Create a LinkedIn post..."
3. **Format:** "Max 200 words, include hook..."
4. **Input:** "{content}"

This ensures consistent, quality outputs.

---

## 5. CHALLENGES & SOLUTIONS

### Challenge 1: API Rate Limits

**Problem:**

- Free tier: 10 requests/min, 50/day
- Large articles need 30+ API calls
- Quickly hit quota

**Solution:**

- Added 6-second delays between calls
- Trim content to 3,000 characters
- Created simplified version (4 calls only)

## Challenge 2: Web Scraping Blocked

**Problem:**

- NYTimes, WSJ block scrapers (403 errors)
- Paywalled content inaccessible

**Solution:**

- Added robust error handling
- Recommend Wikipedia, BBC, tech blogs
- Added PDF and text file alternatives

**Sites that work:**

- ✅ Wikipedia
- ✅ BBC News
- ✅ Tech blogs (TechCrunch, The Verge)

## Challenge 3: YouTube Transcripts

**Problem:**

- Not all videos have transcripts
- API version conflicts

**Solution:**

- Pivoted to text file input for demo
- YouTube as "bonus feature"
- Focus on reliable URL extraction

## Challenge 4: Content Length

**Problem:**

- Wikipedia: 100,000+ characters
- Exceeds API limits

**Solution:**

- Truncate to first 3,000 characters
- Preserves intro + key points
- Stays within quota

---

# 6. POSSIBLE IMPROVEMENTS

### Short-Term (1-2 weeks)

- Add Instagram captions with hashtags
- Add email newsletter format
- Add blog post outlines
- Keyword extraction
- Sentiment analysis

### Medium-Term (1-3 months)

- Web interface (Streamlit)
- Save content history
- Multi-language support
- A/B test different prompts
- Analytics dashboard

### Long-Term (3-6 months)

- Direct posting to social media
- Engagement tracking
- Team collaboration features
- Image generation for posts
- Video script to actual video

---

# 7. USE CASES

### Content Marketing Team

- Publish 5 blogs/week
- Generate social content in 5 min vs 2 hours
- **95% time savings**

### YouTuber

- Promote videos across platforms
- One command generates all formats
- 10x content distribution

### Academic Researcher

- Convert research to public summary
- Social media announcements
- Press releases

---

# 8. SKILLS DEMONSTRATED

## Technical Skills

✅ API integration & authentication
✅ Web scraping with BeautifulSoup
✅ Natural Language Processing
✅ Prompt engineering
✅ Error handling & rate limiting
✅ Command-line tools (argparse)
✅ Python development

## AI/ML Concepts

✅ Large Language Models (LLMs)
✅ Token management
✅ Context optimization
✅ AI agent design patterns

## Software Engineering

✅ Modular code architecture
✅ Environment variable security
✅ Documentation
✅ Version control (Git)

---

# 9. CONCLUSION

This AI Content Repurposing Agent demonstrates practical AI automation for marketing. It successfully:

✅ **Solves Real Problem:** Automates hours of manual work
✅ **Uses Modern AI:** Google Gemini API
✅ **Production-Ready:** Error handling, rate limiting
✅ **Scalable:** Easily add more formats
✅ **Portfolio-Worthy:** Shows technical + business skills

## Key Learnings

1. AI augments human creativity
2. Prompt engineering is critical
3. Always handle edge cases
4. User experience matters
5. Iterate based on feedback

### Next Steps

If continuing:

1. Build Streamlit web interface
2. Add database for history
3. Implement direct social posting
4. Create analytics dashboard
5. Multi-language support

---

## 10. PROJECT INFORMATION

**Deliverables Completed:**

- ✅ AI Agent Code (Python)
- ✅ Demo Video (1-2 minutes)
- ✅ Documentation (This file)

**Student:** Sai Prasad

---

**This project demonstrates practical AI automation skills and understanding of content marketing workflows.**