YouTube Video Downloader API Enhancement Report

This report outlines the plan to enhance our existing YouTube video downloader API, which is currently hosted on RapidAPI and uses yt-dlp. The goal is to expand the API's capabilities by including additional metadata about the videos being processed.

 **by Muhammad Haroon**

Current Functionality

Our API currently provides:

- Video download links
- Audio download links
- Subtitles

Proposed Enhancements

We aim to include the following additional metadata for each video:

1. Video ID
2. Title
3. Length in seconds
4. Channel title
5. Channel ID
6. Description
7. Ratings allowance
8. View count
9. Privacy status
10. Unplugged corpus status
11. Live content status
12. Crawlability status
13. Family-safe status
14. Available countries
15. Unlisted status
16. Category
17. Publish date
18. Published at timestamp
19. Upload date

Implementation Plan

The implementation plan involves several key steps:

1. ## Update yt-dlp Extraction

   Modify the yt-dlp extraction process to retrieve the additional metadata. Most of these fields are already available through yt-dlp's info extraction.

2. ## Map yt-dlp Fields to API Response

   Create a mapping function to transform yt-dlp's output to our API's response format.

3. ## Integrate with Existing API

   Modify the main API function to include the new metadata.

4. ## Error Handling and Validation

   Implement robust error handling and input validation.

5. ## API Documentation Update

   Update the API documentation on RapidAPI to reflect the new metadata fields and their descriptions.

6. ## Testing

   Develop a comprehensive test suite.

7. ## Performance Optimization

   Monitor API performance after adding new fields. If necessary, implement caching mechanisms or optimize the extraction process.

yt-dlp Extraction

The yt-dlp extraction process will be updated to retrieve the additional metadata. This involves modifying the existing code to include the necessary fields. The following code snippet demonstrates how to extract video information using yt-dlp:

```
import yt_dlp
def extract_video_info(video_url):
    ydl_opts = {'skip_download': True}
    with yt_dlp.YoutubeDL(ydl_opts) as ydl:
        info = ydl.extract_info(video_url, download=False)
        return info
```

Mapping yt-dlp Fields to API Response

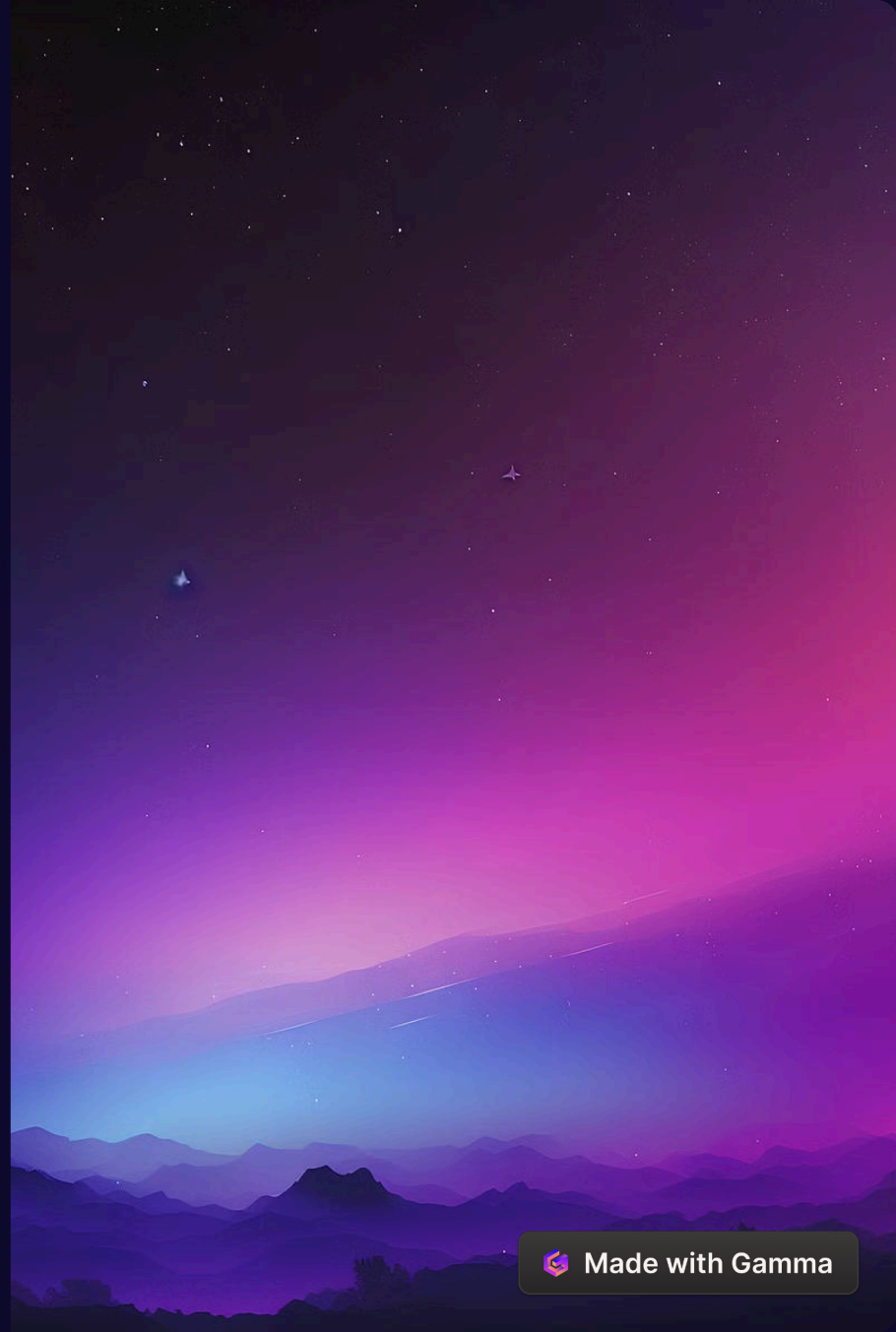
A mapping function will be created to transform the output from yt-dlp into the format required for our API response. This function will map the relevant fields from yt-dlp's output to the corresponding fields in our API response. The following code snippet demonstrates the mapping function:

```
def map_to_api_response(yt_dlp_info):
    return {
        "id": yt_dlp_info.get('id'),
        "title": yt_dlp_info.get('title'),
        "lengthSeconds": yt_dlp_info.get('duration'),
        "channelTitle": yt_dlp_info.get('channel'),
        "channelId": yt_dlp_info.get('channel_id'),
        "description": yt_dlp_info.get('description'),
        "allowRatings": yt_dlp_info.get('allow_ratings', False),
        "viewCount": yt_dlp_info.get('view_count'),
        "isPrivate": yt_dlp_info.get('availability') == 'private',
        "isUnpluggedCorpus": yt_dlp_info.get('is_unplugged_corpus', False),
        "isLiveContent": yt_dlp_info.get('is_live', False),
        "isCrawlable": yt_dlp_info.get('webpage_url_basename') == 'watch',
        "isFamilySafe": yt_dlp_info.get('age_limit', 0) == 0,
        "availableCountries": yt_dlp_info.get('country_list', []),
        "isUnlisted": yt_dlp_info.get('availability') == 'unlisted',
        "category": yt_dlp_info.get('category'),
        "publishDate": yt_dlp_info.get('upload_date'),
        "publishedAt": yt_dlp_info.get('release_timestamp'),
        "uploadDate": yt_dlp_info.get('upload_date')
    }
```

API Integration

The new metadata will be integrated into the existing API function. This involves modifying the function to include the metadata retrieval and mapping steps. The following code snippet demonstrates the updated API function:

```
def get_video_info(video_url):
    yt_dlp_info = extract_video_info(video_url)
    metadata = map_to_api_response(yt_dlp_info)
    # Existing functionality
    download_links = get_download_links(yt_dlp_info)
    subtitles = get_subtitles(yt_dlp_info)
    return {
        "metadata": metadata,
        "video_download_links": download_links['video'],
        "audio_download_links": download_links['audio'],
        "subtitles": subtitles
    }
```



Conclusion

This enhancement will significantly increase the value of our YouTube video downloader API by providing comprehensive metadata along with the existing download links and subtitles. It will cater to a wider range of use cases and potentially attract more users to our RapidAPI offering.