Rabyte Translation API Documentation

Overview

The Rabyte Translation API provides an interface for translating video files with optional profanity removal and language selection. This document outlines the available endpoints, request parameters, and potential responses for future enhancements and integrations.

Endpoints

Health Check

GET /healthcheck

Check the health status of the API.

Request

- No parameters.

Response

- **200 OK**: { "status": "healthy" }
- **500 Internal Server Error**: { "status": "failure" }

Example

```
GET /healthcheck HTTP/1.1
Host: <your-domain>
```

Response

```
Json:
  Copy code
  {
    "status": "healthy"
  }
```

Video Translation

POST /rabyte-translation

Translate a provided video file with optional settings.

Request

- **Headers**: Content-Type: multipart/form-data
- **Body**:
  - file: The video file to be translated (required, must be .mp4).

    - remove_profanity: Flag to indicate whether to remove profanity from the translation (optional, default: No).
    - language: The target language for translation (optional, default: English).

Response

- **200 OK**: The translated video file is returned as an attachment.
- **400 Bad Request**: Error messages for invalid requests.
    - No file part in the request
    - No selected file
    - Invalid file format
- **500 Internal Server Error**: Error occurred

Response

- **On Success**: The translated video file is returned as an attachment.
- **On Failure**:

```
Json:
  Copy code
  {
    "error": "No file part in the request"
  }
```

Error Handling

The API handles errors gracefully and returns appropriate HTTP status codes along with error messages to help diagnose issues.

- **400 Bad Request**: Indicates issues with the client's request, such as missing file part or invalid file format.
- **500 Internal Server Error**: Indicates a problem on the server side, such as unexpected errors during processing.

Future Enhancements

For future integrations and enhancements, consider the following features:

- **Support for Multiple File Formats**: Extend support to other video formats such as .avi, .mkv, etc.
- **Asynchronous Processing**: Implement asynchronous processing for large files and provide a callback mechanism or a status endpoint to check the progress.
- **Authentication**: Secure the endpoints with authentication and authorization mechanisms.
- **Detailed Logging and Monitoring**: Improve error logging and monitoring for better diagnostics and alerting.