PROJECT DEVELOPMENT PHASE

Exception Handling

Date	4 November 2023
Team ID	NM2023TMID10148
Project Name	How to create a Brand Promo Video using Canva.

Exception handling is a crucial aspect of software development to ensure that your code can gracefully handle unexpected situations and errors. When creating a promo video using a service like Canva through programming, there are various potential exceptions you should consider. Below is a general outline of how you can handle exceptions in such a scenario:

API Access:

Ensure that you have the necessary API keys or access credentials to use Canva's services.

Use proper exception handling to catch authentication errors if your access credentials are incorrect.

API Requests:

When making requests to Canva's API, handle potential network-related exceptions, such as network timeouts or connectivity issues.

Implement error handling to manage unexpected response statuses from the API.

File Handling:

If you are dealing with video or image files in your code, handle exceptions related to file I/O, such as file not found, permission issues, or file format errors.

Check for issues like insufficient disk space if you are saving files locally.

Rate Limiting:

Canva's API may have rate limits to prevent abuse. Implement exception handling to manage rate-limiting errors and retry logic when appropriate.

Invalid Input:

Validate user input to ensure it meets Canva's requirements, such as image dimensions, file formats, or video specifications. Handle exceptions for invalid inputs.

Resource Availability:

Check for the availability of resources like fonts, templates, or design elements before using them in your promo video.

Billing and Subscription:

Handle exceptions related to billing or subscription issues, such as exceeding usage limits or payment failures.

Logging and Reporting:

Implement a logging mechanism to record errors and exceptions. This will help in debugging and identifying issues promptly.

Consider sending notifications or alerts for critical errors so that you can address them in a timely manner.

Graceful Degradation:

When errors occur, strive to provide a graceful degradation of service, allowing the application to continue functioning in some capacity, even if some features are unavailable.

User-Friendly Messages:

Display user-friendly error messages and guidance on how to resolve issues when exceptions occur. Avoid exposing technical error details to end-users.

```
import requests

try:
    # Make a request to Canva's API
    response = requests.get('https://api.canva.com/some/endpoint')
```

```
    # Check for API response errors
    response.raise_for_status()
  # Process the response
    process_response(response)
except requests.exceptions.RequestException as e:
    # Handle network-related errors
    print(f"Network error: {e}")
except requests.exceptions.HTTPError as e:
    # Handle HTTP errors (e.g., 4xx, 5xx status codes)
    print(f"HTTP error: {e}")
except Exception as e:
    # Handle other unexpected exceptions
    print(f"An unexpected error occurred: {e}")
```