Vision Statement: YouTube Summary and Feedback System

Vision

We aim to create a tool that leverages large language models to provide actionable insights about YouTube videos and audience engagement. This system will provide video summaries, sentiment breakdowns, and targeted feedback extraction.

Target Users

- **Primary Users:** YouTube creators seeking quick, structured insights to improve video engagement and content strategy.
- **Secondary Users:** Educators, and businesses using YouTube for content outreach.

System Components

1. Data Retrieval & Preprocessing

Purpose: Gather essential data—video transcripts and viewer comments—for meaningful analysis while ensuring data quality.

- **YouTube API Integration:** Extracts video metadata, transcripts, and comments while managing API request limits.
- **Preprocessing Pipeline:**
    - Cleans transcript data by removing timestamps, filler words, and extraneous content.
    - Filters out irrelevant comments before any further processing, including:
        - Spam & Bot Comments (e.g., “Check out my channel!”)
        - Off-Topic Discussions (e.g., “This reminds me of my childhood.”)
        - Self-Promotion (e.g., links to unrelated content or advertisements)
        - Unrelated Conversations (e.g., personal arguments or jokes not related to the video)
- **Data Storage:** Saves clean transcripts and relevant comments.
- **Operational Constraints:**
    - Videos longer than 1 hour are excluded due to the generation of larger transcripts as time increases.
    - Minimum 100 comments required to ensure a balanced sentiment analysis and constructive feedback.

2. Video Summarization

Purpose: Condense lengthy video transcripts into summaries.

- **Summarization Engine:**

- Uses ChatGPT API to generate key takeaways.
- Segments transcript intelligently adhere to API processing limits.

❖ **Format Handling:** Output summaries in paragraphs for clarity.

3. Sentiment Analysis

Purpose: Classify audience sentiment to understand viewer engagement and reactions.

❖ **Sentiment Classification:**

- Uses VADER (Valence Aware Dictionary and sEntiment Reasoner) for fast, reliable scoring.
- Labels comments as positive, negative, or neutral based on textual sentiment patterns.

4. Constructive Feedback Extraction

Purpose: Identify patterns in comments to highlight content strengths and areas needing improvement.

❖ **ChatGPT API for Pros & Cons Analysis:**

- Highlights specific content strengths (e.g., “Great storytelling” or “Excellent editing”).
- Identifies areas for improvement (e.g., “Audio quality needs enhancement” or “More concise explanations needed”).

❖ **Structured Report Generation:**

- Provides a “What’s Working” vs. “Needs Improvement” section to streamline creator insights.

5. User Interface & Report Generation

Purpose: Home page as webpage 1 and present results including video summary, sentiment analysis and constructive feedback(strengths and weaknesses) in webpage 2.

❖ **Home Page:** Allows users to paste a YouTube URL and initiate analysis with minimal effort.

❖ **Results Presentation:** Provides structured summaries, sentiment visuals, and feedback highlights in a clean layout in a new webpage.

Competitive Advantage

Most existing tools—like YouTube Studio, VidIQ, and TubeBuddy—focus on SEO metrics and engagement analytics but lack deep audience insight generation. This system goes further by offering:

❖ AI-powered video summarization.
❖ Structured audience sentiment analysis beyond basic engagement metrics.

- ChatGPT-based feedback extraction for direct, actionable insights.
- Automated irrelevant comment filtering to ensure meaningful data analysis.
- Designed to be achievable within a semester, with a phased feature rollout.

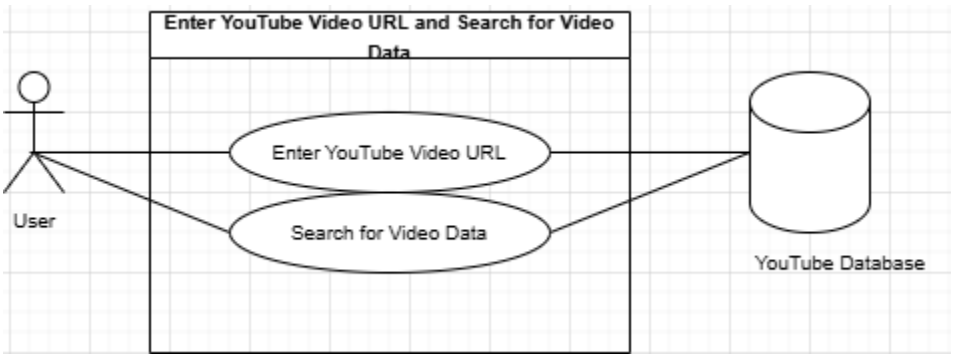
ID	Product Backlog	Estimation	Priority
1	As a User, I can enter a YouTube video URL on the home page and click “Search” to initiate video data retrieval (metadata including title, description and duration), transcript, and comments).	3	1
2	The Web Server can validate the YouTube URL format and extract the video ID.	2	2
3	The Web Server can retrieve video metadata (e.g., title, description, duration) from the YouTube API using the video ID.	3	3
4	The Web Server can retrieve the video transcript from the YouTube API using the video ID.	3	4
5	The Web Server can retrieve at least 100 video comments from the YouTube API using the video ID.	2	5
6	The Web Server will enforce operational constraints by excluding videos longer than 1 hour and ensuring at least 100 comments are available; if not, it notifies the User.	2	6
7	The Web Server can remove timestamps from the retrieved transcript.	1	7
8	The Web Server can remove filler words from the retrieved transcript.	1	8
9	The Web Server can remove spam content from the retrieved transcript.	2	9
10	The Web Server can remove off-topic remarks from the retrieved transcript.	2	10
11	The Web Server can remove unrelated discussions from the retrieved transcript.	2	11
12	The Web Server can process the cleaned transcript for further analysis.	1	12
13	The Web Server can store the processed transcript data.	1	13
14	The Web Server can send the clean transcript to the ChatGPT API to generate a concise video summary in paragraph form and save it.	5	14
15	VADER sentiment analysis library from Python can define and implement the interface.	2	15
16	The Web Server will use VADER from Python library to classify filtered comments as positive, negative, or neutral, and store the sentiment analysis results.	3	16
17	The ChatGPT API can define and implement the interface for processing filtered comments.	1	17
18	The ChatGPT API can extract actionable “What’s Working” feedback from filtered comments.	1	18
19	The ChatGPT API can extract actionable “Needs Improvement” feedback from filtered comments.	1	19
20	The Web Server can generate a structured feedback report categorizing “What’s Working” and “Needs Improvement” and store the report.	2	20
21	As a User, I can view a results page that displays the generated video summary.	3	21

22	As a User, I can view a results page that displays detailed sentiment analysis results of comments.	3	22
23	As a User, I can view a results page that displays the structured feedback report categorizing “What’s Working” and “Needs Improvement”.	4	23

Estimated: 50 days

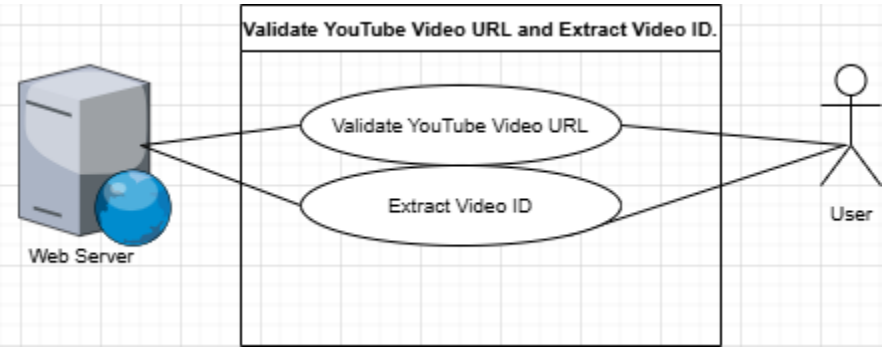
Use Case: Enter YouTube Video URL and Search

Use Case Name	Enter YouTube Video URL and Search
Reference to Product Backlog	PB-1
Scope	YouTube Video Analysis System
Level	User-goal
Primary Actor	User
Stakeholders and Their Interests	User: Needs an uncomplicated method to begin video analysis by entering a URL System: Relies on receiving a properly formatted URL to proceed
Preconditions	User is on the home page with a clearly visible YouTube URL input field network connection is stable
Success Criteria	The system accepts the URL and submits it for further processing
Main Success Scenario	1. User navigates to the home page 2. User enters a YouTube video URL 3. User clicks “Search” 4. The URL is submitted to the system
Extensions	If no URL is entered, display “URL required” if the URL appears malformed, prompt the user to correct it
Special Requirements	Input must strictly follow standard YouTube URL format
Technology and Data Variations	Web-based interface accessible via common browsers URL data is in text format
Frequency	Each time a user presses a search button
Open Issues	None



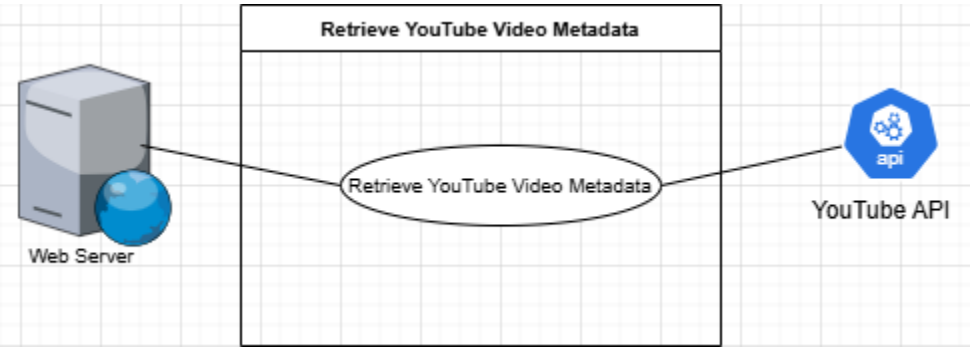
Use Case: Validate YouTube Video URL and Extract Video ID

Use Case Name	Validate YouTube Video URL and Extract Video ID
Reference to Product Backlog	PB-2
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Expects analysis to proceed only with a valid URL System: Requires a correct video ID to request video data
Preconditions	A YouTube video URL is received from the user (via UC-1) network connection is stable
Success Criteria	The system confirms the URL is valid and successfully extracts the video ID
Main Success Scenario	1. Web Server receives the URL from the user 2. The system verifies the URL format against YouTube standards 3. If valid, the system extracts the video ID 4. The extracted video ID is used to request data from the YouTube API
Extensions	If validation fails, display “Invalid URL format. Please enter a valid YouTube video URL” and prompt re-entry if extraction fails, display an error
Special Requirements	URL must exactly match YouTube’s recognized format
Technology and Data Variations	Data is received and processed in JSON format from the API
Frequency	Each URL submission
Open Issues	None



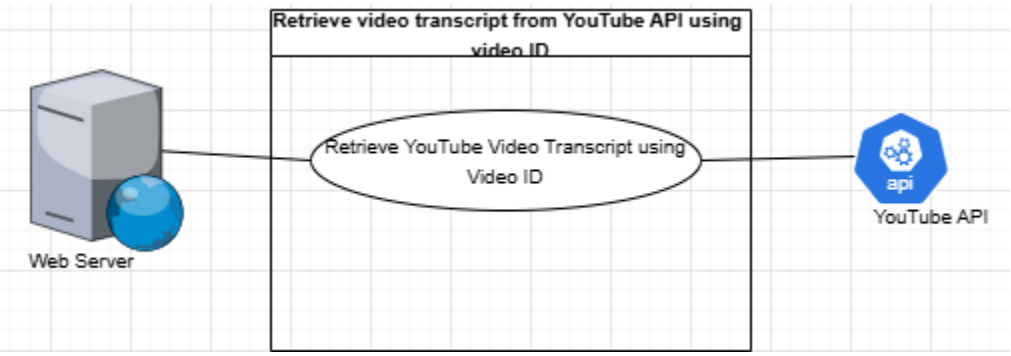
Use Case: Retrieve YouTube Video Metadata

Use Case Name	Retrieve YouTube Video Metadata
Reference to Product Backlog	PB-3
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Needs accurate video details (title, description, duration) System: Must retrieve complete metadata to support subsequent analysis
Preconditions	A valid video ID is available from UC-2 network connection is stable
Success Criteria	The system retrieves complete video metadata including title, description, and duration
Main Success Scenario	1. Web Server sends a request to the YouTube API using the video ID 2. API returns the video metadata 3. The system parses the metadata for use in analysis
Extensions	If the API call fails or metadata is incomplete, display an error
Special Requirements	Metadata must include title, description, and duration
Technology and Data Variations	JSON format from the YouTube API
Frequency	For every valid video URL processed
Open Issues	None



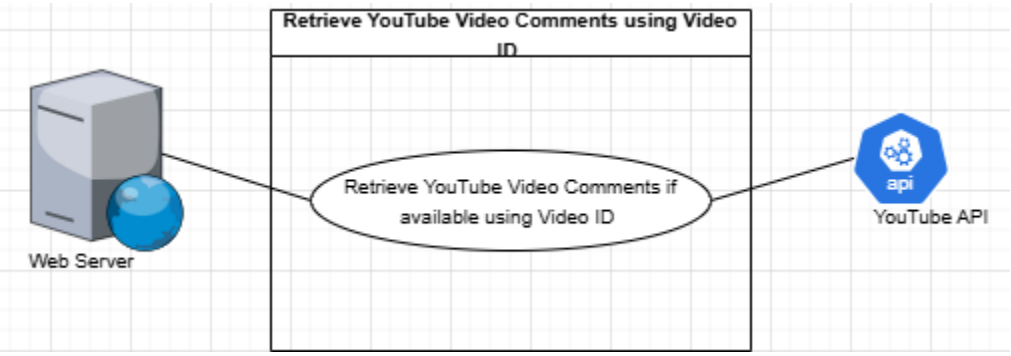
Use Case: Retrieve Video Transcript from YouTube API using Video ID

Use Case Name	Retrieve Video Transcript from YouTube API
Reference to Product Backlog	PB-4
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Requires the transcript text to understand the video content System: Needs the transcript for real-time analysis
Preconditions	A valid video ID is available network connection is stable
Success Criteria	The system retrieves the full transcript in plain text
Main Success Scenario	1. Web Server requests the transcript from the YouTube API using the video ID 2. API returns the transcript 3. The system obtains the transcript for analysis
Extensions	If the transcript is unavailable or the API fails, display an error
Special Requirements	Transcript must be provided in plain text
Technology and Data Variations	Transcript data may be delivered as plain text or in JSON
Frequency	For each video that includes a transcript
Open Issues	None



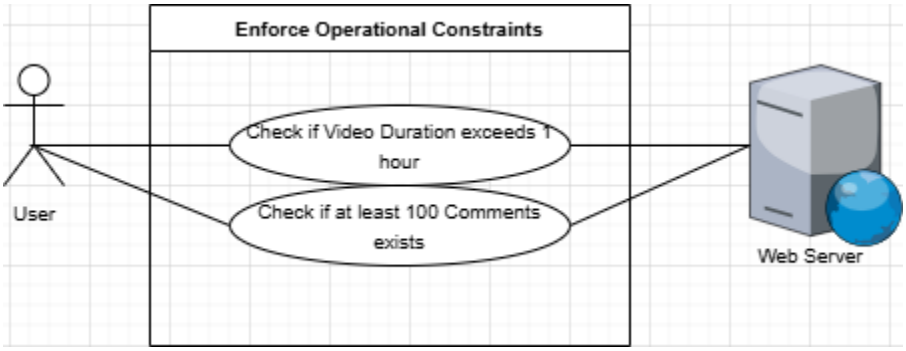
Use Case: Retrieve YouTube Video Comments

Use Case Name	Retrieve YouTube Video Comments from YouTube API
Reference to Product Backlog	PB-5
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Requires a sufficient number of comments to ensure meaningful analysis System: Must retrieve at least 100 comments
Preconditions	A valid video ID is available network connection is stable
Success Criteria	The system retrieves a minimum of 100 comments
Main Success Scenario	1. Web Server sends a comment request to the YouTube API using the video ID 2. API returns a list of comments 3. The system confirms that at least 100 comments have been retrieved
Extensions	If fewer than 100 comments are returned, display "**Insufficient comments. Please select a video with at least 100 comments**"
Special Requirements	Must retrieve a minimum of 100 comments
Technology and Data Variations	Comments are returned in JSON (possibly paginated)
Frequency	For each valid video URL processed
Open Issues	None



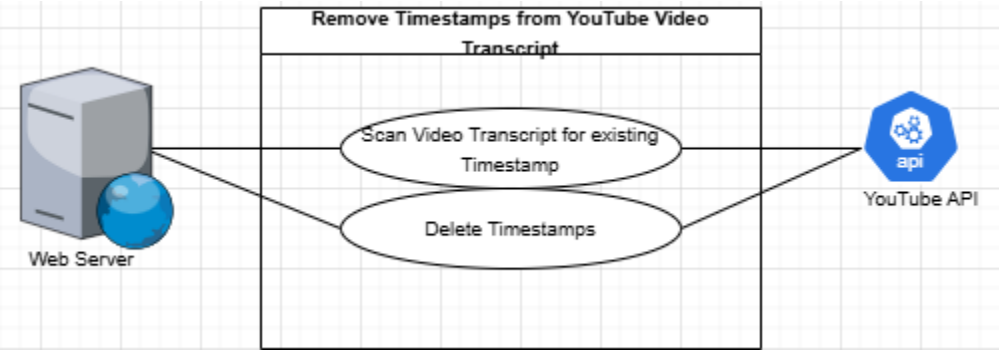
Use Case: Enforce Operational Constraints

Use Case Name	Enforce Operational Constraints
Reference to Product Backlog	PB-6
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Expects that only videos meeting specific criteria are analyzed System: Must enforce video duration and comment count limits to ensure processing quality
Preconditions	Video metadata and comment data are available from UC-3 and UC-5 network connection is stable
Success Criteria	The video's duration is 60 minutes or less and the comment count is 100 or more
Main Success Scenario	1. Web Server reads the video duration from metadata 2. Checks that the duration is ≤ 60 minutes 3. Counts the comments 4. If comments are 100 or more, the video is approved for analysis
Extensions	If the video duration exceeds 60 minutes, display "Video duration exceeds the limit. Please select a shorter video" if comments are fewer than 100, display "Not enough comments. Please select a different video"
Special Requirements	Duration must be ≤ 60 minutes comment count must be ≥ 100
Technology and Data Variations	Numerical data obtained from API responses
Frequency	Each time a video is submitted for analysis
Open Issues	None



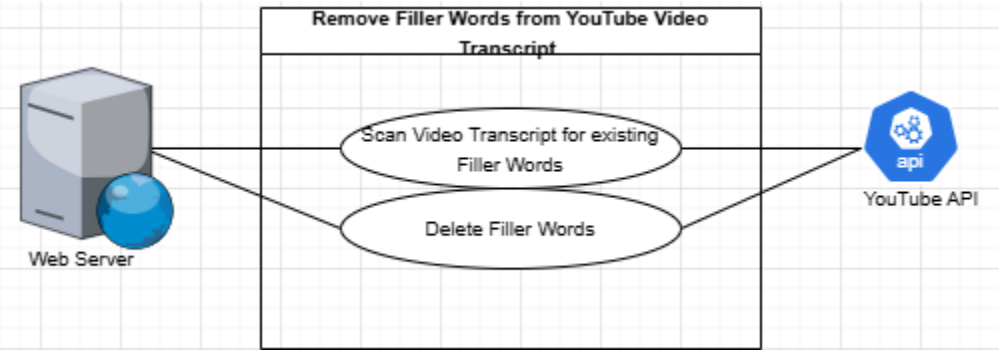
Use Case: Remove Timestamps from Transcript

Use Case Name	Remove Timestamps from Transcript
Reference to Product Backlog	PB-7
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Expects a transcript without distracting time markers System: Must eliminate all timestamp patterns to clarify content
Preconditions	A raw transcript is received network connection is stable
Success Criteria	The transcript is provided with all timestamp markers removed
Main Success Scenario	1. Web Server receives the raw transcript 2. Identifies standard timestamp patterns (e.g., “00:01:23”) 3. Removes these patterns from the transcript
Extensions	If any timestamps remain due to non-standard formatting, display an error
Special Requirements	Removal must target only standard timestamp formats without affecting the dialogue
Technology and Data Variations	Utilizes regular expressions to detect timestamps
Frequency	Each time a transcript is processed
Open Issues	None



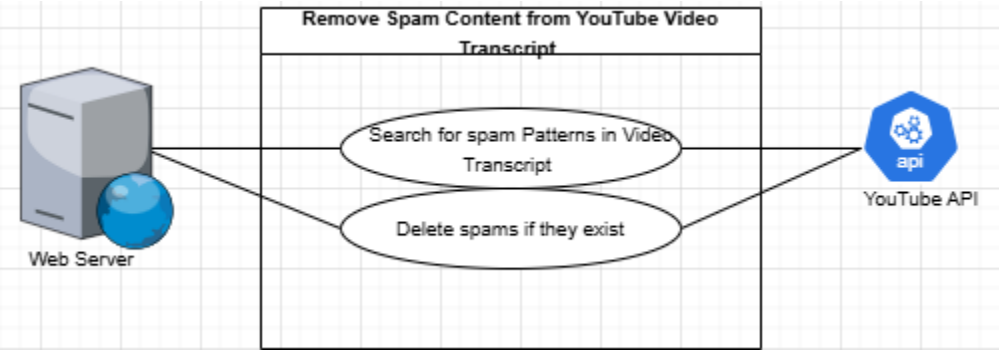
Use Case: Remove Filler Words from Transcript

Use Case Name	Remove Filler Words from Transcript
Reference to Product Backlog	PB-8
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Requires a transcript that is free of non-essential verbal fillers System: Must remove a specified set of filler words to enhance clarity
Preconditions	A raw transcript is received network connection is stable
Success Criteria	The transcript is presented without any filler words (e.g., “um”, “uh”)
Main Success Scenario	1. Web Server receives the raw transcript 2. Identifies filler words based on a predetermined list 3. Removes all occurrences of these words
Extensions	If any filler words remain, display an error
Special Requirements	Only the words in the specified list are to be removed without altering the meaning
Technology and Data Variations	Uses a text search algorithm based on a fixed list
Frequency	Each time a transcript is processed
Open Issues	None



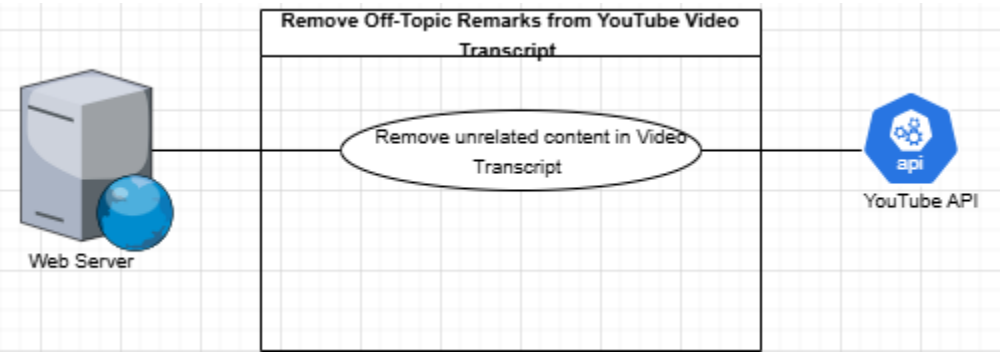
Use Case: Remove Spam Content from Transcript

Use Case Name	Remove Spam Content from Transcript
Reference to Product Backlog	PB-9
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Requires a transcript devoid of promotional or repetitive content System: Must remove text that qualifies as spam according to predefined criteria
Preconditions	A raw transcript is received network connection is stable
Success Criteria	The transcript is cleansed of all content identified as spam
Main Success Scenario	1. Web Server receives the raw transcript 2. Scans the text for content matching the spam criteria 3. Removes the identified spam segments
Extensions	If spam is not entirely removed, display an error
Special Requirements	Removal must target only text meeting the established spam rules
Technology and Data Variations	Uses text matching algorithms
Frequency	Each time a transcript is processed
Open Issues	None



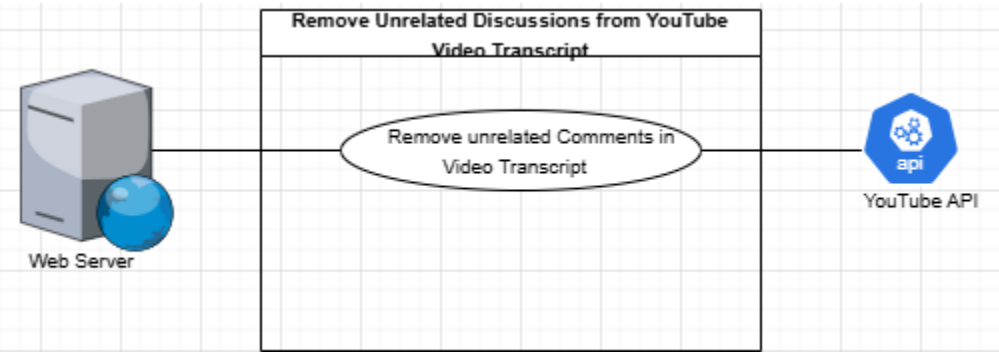
Use Case: Remove Off-Topic Remarks from Transcript

Use Case Name	Remove Off-Topic Remarks from Transcript
Reference to Product Backlog	PB-10
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Expects a transcript that focuses solely on the video's subject matter System: Must remove comments that deviates from the core topic
Preconditions	A raw transcript is received network connection is stable
Success Criteria	The transcript is delivered without any off-topic remarks
Main Success Scenario	1. Web Server receives the raw transcript 2. Identifies segments that do not relate to the main video topic 3. Removes these segments
Extensions	If off-topic content remains, display an error
Special Requirements	Removal must target only non-relevant dialogue
Technology and Data Variations	Uses keyword matching methods
Frequency	Each time a transcript is processed
Open Issues	None



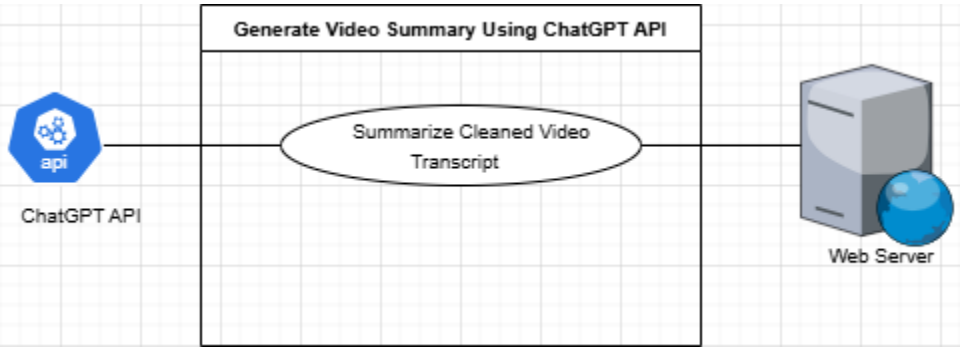
Use Case: Remove Unrelated Discussions from Transcript

Use Case Name	Remove Unrelated Discussions from Transcript
Reference to Product Backlog	PB-11
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Expects a transcript that reflects only the video content System: Must eliminate discussions that are not directly related to the video subject
Preconditions	A raw transcript is received network connection is stable
Success Criteria	The transcript is free from discussions unrelated to the video's subject
Main Success Scenario	1. Web Server receives the raw transcript 2. Identifies segments that are not directly related to the video content 3. Removes these unrelated segments
Extensions	If unrelated discussions persist, display an error
Special Requirements	Removal must target only off-topic discussions
Technology and Data Variations	Uses semantic filtering with predefined criteria
Frequency	Each time a transcript is processed
Open Issues	None



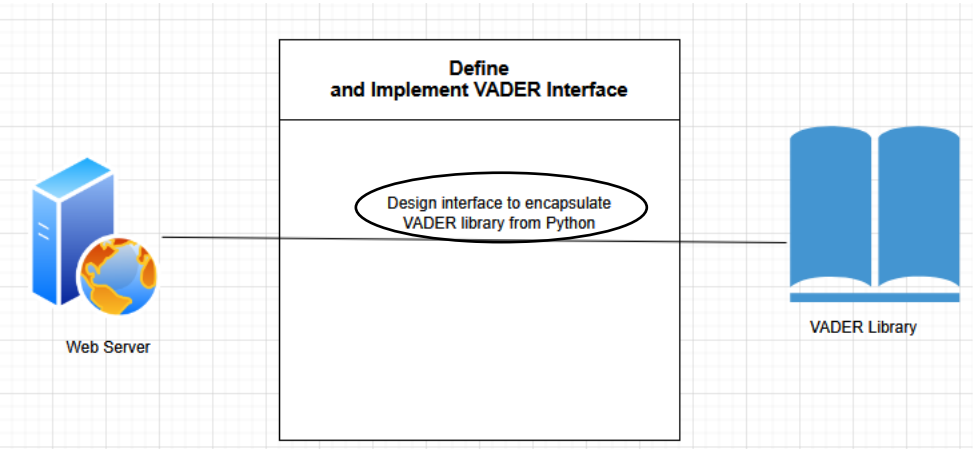
Use Case: Generate Video Summary Using ChatGPT API

Use Case Name	Generate Video Summary Using ChatGPT API
Reference to Product Backlog	PB-14
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Expects a brief and clear summary of the video content System: Must generate the summary on the fly using the clean transcript
Preconditions	A clean transcript is available from previous processing network connection is stable
Success Criteria	A concise, one-paragraph summary is produced and presented
Main Success Scenario	1. Web Server sends the clean transcript to the ChatGPT API 2. ChatGPT API processes the transcript and returns a summary focusing on key topics 3. Web Server displays the summary on the results page
Extensions	If the API call fails, display “Failed to generate summary. Please try again later”
Special Requirements	The summary must be confined to one paragraph and contain only essential information
Technology and Data Variations	None
Frequency	Every time a clean transcript is processed for summarization
Open Issues	None



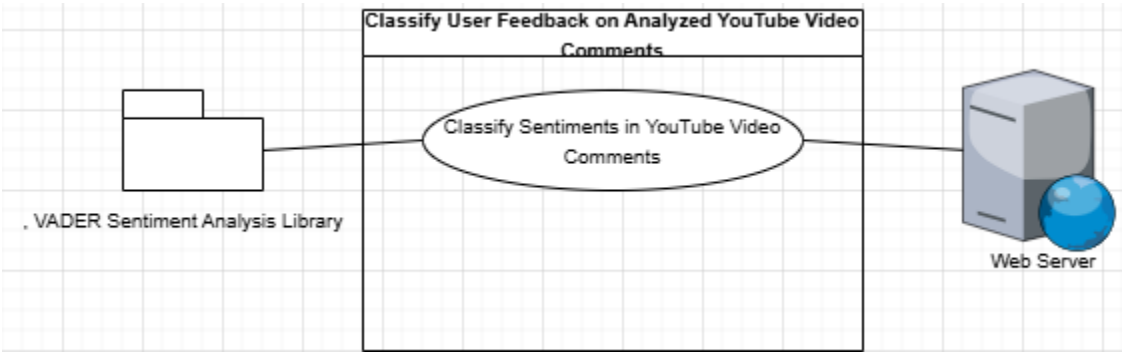
Use Case: Define and Implement VADER Interface

Use Case Name	Define and Implement VADER Interface
Reference to Product Backlog	PB-15
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	System: Requires a dedicated interface for integrating the VADER sentiment analysis library to evaluate comment sentiment
Preconditions	Sentiment analysis requirements are documented network connection is stable
Success Criteria	A VADER interface is implemented and successfully returns sentiment scores for sample text
Main Success Scenario	1. Review VADER library documentation 2. Design an interface encapsulating VADER's functions 3. Implement the interface in the Web Server 4. Validate the interface using sample text inputs
Extensions	If implementation fails, log errors and refine the design until tests pass
Special Requirements	Interface must conform exactly to VADER library specifications
Technology and Data Variations	Developed in Python accepts text input and outputs numerical sentiment scores in JSON
Frequency	Implemented once during system integration maintained thereafter
Open Issues	None



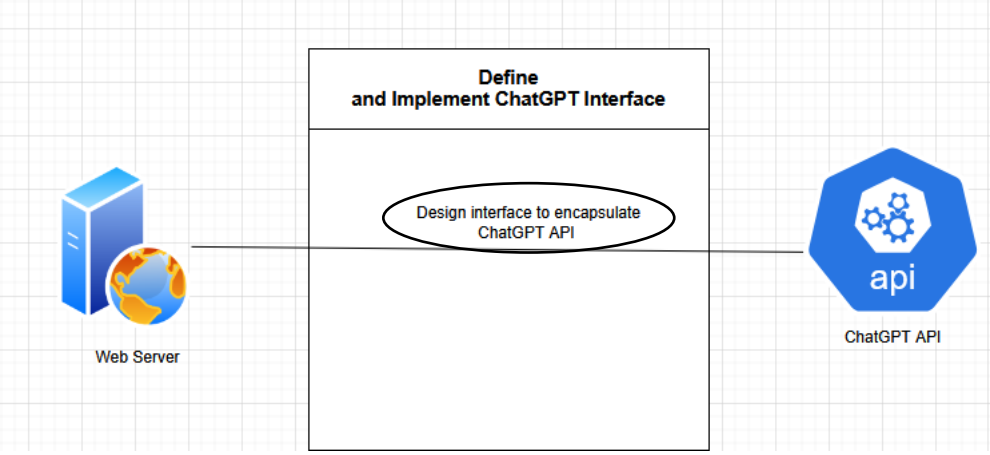
Use Case: Classify Comments with VADER

Use Case Name	Classify Comments with VADER
Reference to Product Backlog	PB-16
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Expects each comment to receive a sentiment label (positive, negative, neutral) System: Must reliably classify comment sentiment
Preconditions	Filtered video comments are available VADER interface is operational network connection is stable
Success Criteria	Every comment is assigned a sentiment classification in real time
Main Success Scenario	1. Web Server iterates over each filtered comment 2. Each comment is submitted to the VADER interface 3. The system receives and records the sentiment classification for each comment
Extensions	If classification fails for any comment, log the error and continue processing remaining comments
Special Requirements	Predefined sentiment thresholds must be applied consistently
Technology and Data Variations	Utilizes the VADER Python library output is in JSON format
Frequency	Each time a new batch of comments is processed
Open Issues	None



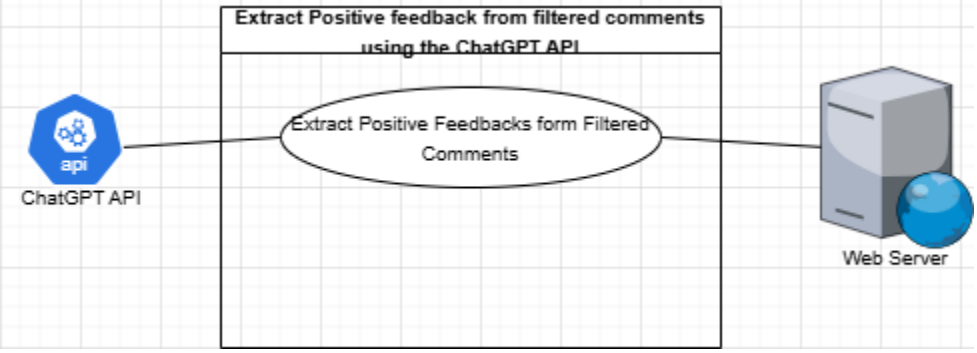
Use Case: Define ChatGPT Interface for Comment Processing

Use Case Name	Define ChatGPT Interface for Comment Processing
Reference to Product Backlog	PB-17
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	System: Requires a robust interface for processing comments via the ChatGPT API for deriving feedback
Preconditions	Comment processing requirements are clearly defined network connection is stable
Success Criteria	The ChatGPT interface is implemented and returns expected outputs for sample comment inputs
Main Success Scenario	1. Review ChatGPT API documentation 2. Design an interface for processing filtered comments 3. Implement the interface within the Web Server 4. Test with sample inputs to verify correct output
Extensions	If testing fails, log errors and adjust the design until successful
Special Requirements	Interface must comply with ChatGPT API specifications
Technology and Data Variations	Uses HTTP API calls with JSON responses
Frequency	Implemented once during integration
Open Issues	None



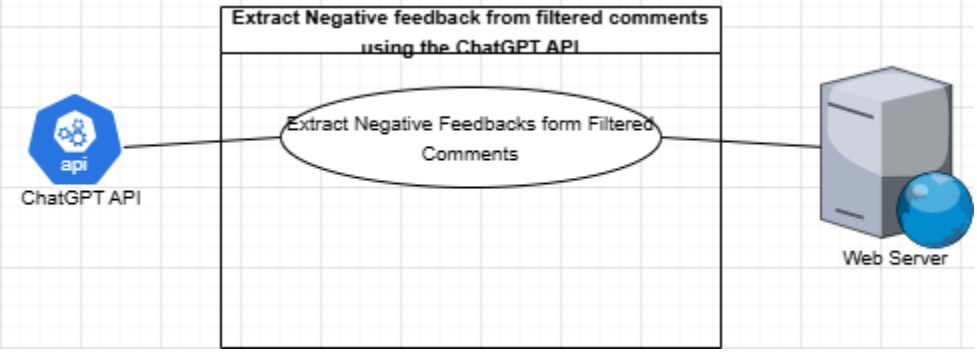
Use Case: Extract “What’s Working” Feedback via ChatGPT

Use Case Name	Extract “What’s Working” Feedback via ChatGPT
Reference to Product Backlog	PB-18
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	ChatGPT API (accessed via Web Server)
Stakeholders and Their Interests	User: Expects to see specific positive aspects from video comments System: Must extract feedback that highlights the strengths reflected in the comments
Preconditions	Filtered comments are available ChatGPT interface is operational network connection is stable
Success Criteria	Positive feedback is extracted and made available for immediate display
Main Success Scenario	1. Web Server sends the filtered comments to the ChatGPT API 2. ChatGPT API processes the comments and extracts feedback on positive aspects 3. The extracted positive feedback is presented on the results page
Extensions	If extraction fails, display an error message
Special Requirements	Feedback must include only positive observations without mixing in neutral or negative remarks
Technology and Data Variations	Uses HTTP API with JSON responses
Frequency	Each time new comment data is processed
Open Issues	None



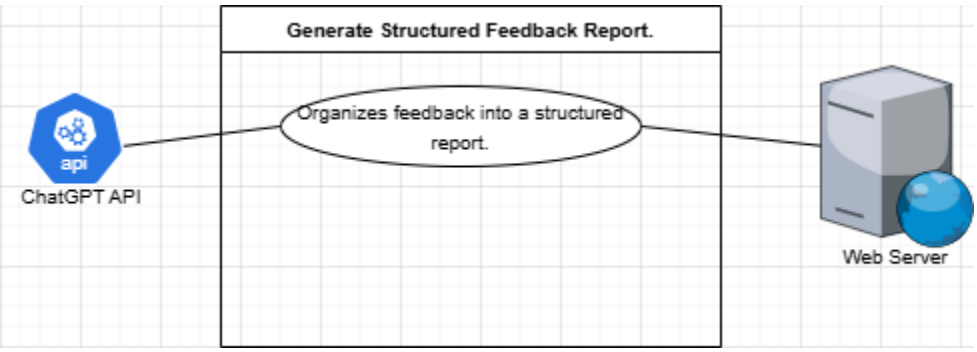
Use Case: Extract “Needs Improvement” Feedback via ChatGPT

Use Case Name	Extract “Needs Improvement” Feedback via ChatGPT
Reference to Product Backlog	PB-19
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	ChatGPT API (accessed via Web Server)
Stakeholders and Their Interests	User: Expects clear suggestions for improvement based on comment analysis System: Must extract feedback that identifies specific areas where the video content can be improved
Preconditions	Filtered comments are available ChatGPT interface is operational network connection is stable
Success Criteria	Feedback focused on improvement areas is extracted and immediately available
Main Success Scenario	1. Web Server sends the filtered comments to the ChatGPT API 2. ChatGPT API processes the comments and extracts feedback on improvement needs 3. The extracted improvement feedback is presented on the results page
Extensions	If extraction fails, display an error message
Special Requirements	Feedback must include only improvement suggestions, clearly separated from positive feedback
Technology and Data Variations	Uses HTTP API with JSON responses
Frequency	Each time new comment data is processed
Open Issues	None



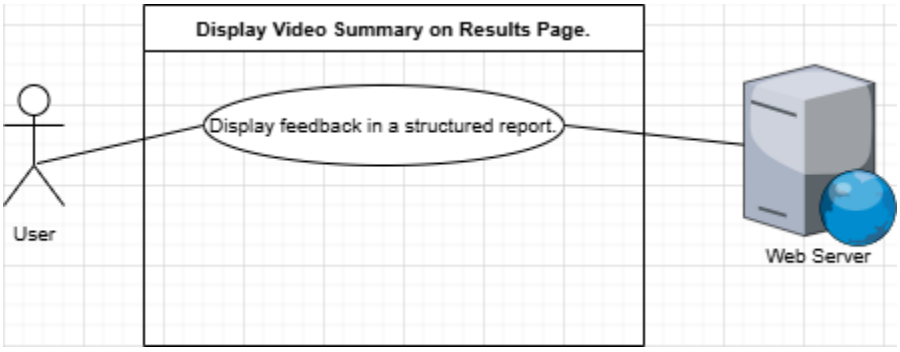
Use Case: Generate Structured Feedback Report

Use Case Name	Generate Structured Feedback Report
Reference to Product Backlog	PB-20
Scope	YouTube Video Analysis System
Level	System-level
Primary Actor	Web Server
Stakeholders and Their Interests	User: Expects a report that clearly distinguishes positive feedback from improvement suggestions System: Must generate the report dynamically and in an organized format
Preconditions	"What's Working" and "Needs Improvement" feedback are available network connection is stable
Success Criteria	A structured report with two clearly defined sections is generated and displayed
Main Success Scenario	1. Web Server retrieves both positive and improvement feedback 2. Organizes the feedback into two separate sections labeled "What's Working" and "Needs Improvement" 3. Displays the report on the results page
Extensions	If report generation fails, display an error message
Special Requirements	The report must clearly separate the two types of feedback without overlap
Technology and Data Variations	Output is rendered in structured text via the web interface
Frequency	Each time comment feedback is processed
Open Issues	None



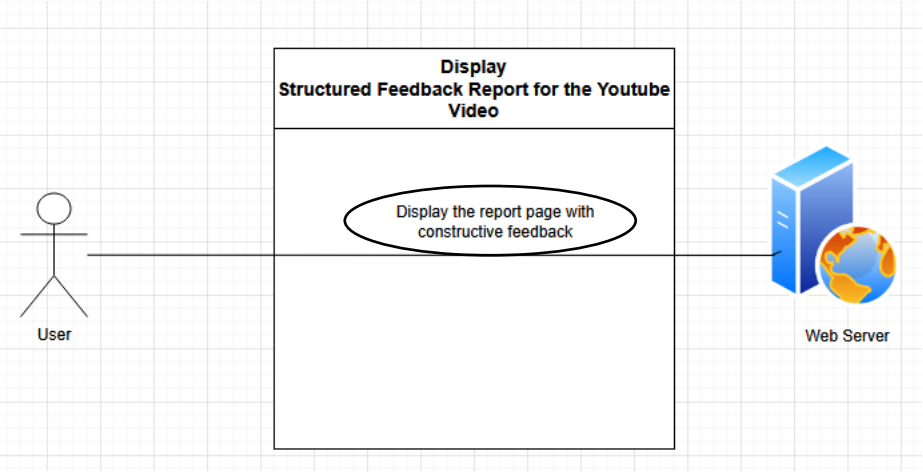
Use Case: Display Video Summary

Use Case Name	Display Video Summary
Reference to Product Backlog	PB-21
Scope	YouTube Video Analysis System
Level	User-goal
Primary Actor	User
Stakeholders and Their Interests	User: Expects a succinct summary of the video content System: Must present the summary clearly and succinctly
Preconditions	Video summary is generated from UC-14 network connection is stable
Success Criteria	Video summary is displayed as a single, clear paragraph on the results page
Main Success Scenario	1. Web Server retrieves the video summary 2. Formats the summary into one clear paragraph 3. Displays the summary on the results page
Extensions	If the summary is not available, display an error message
Special Requirements	Summary must be confined to one paragraph and cover only the key points
Technology and Data Variations	Standard web display format
Frequency	Every time a video is analyzed
Open Issues	None



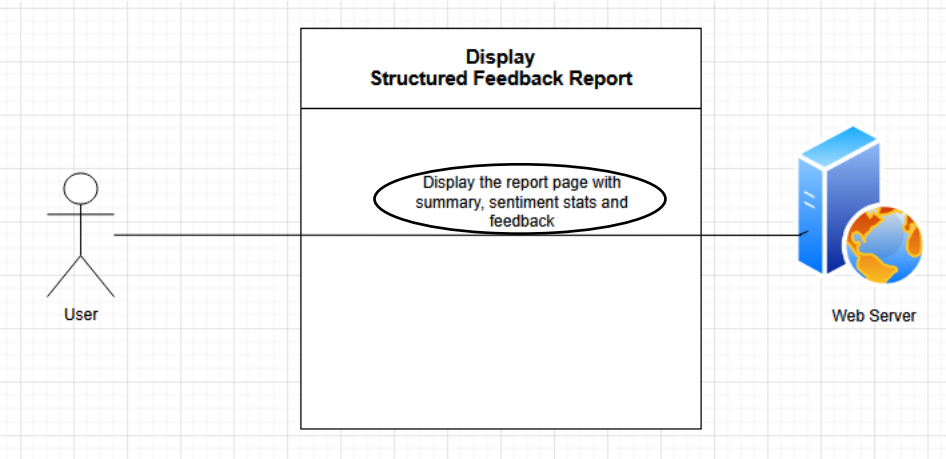
Use Case: Display Sentiment Analysis Results

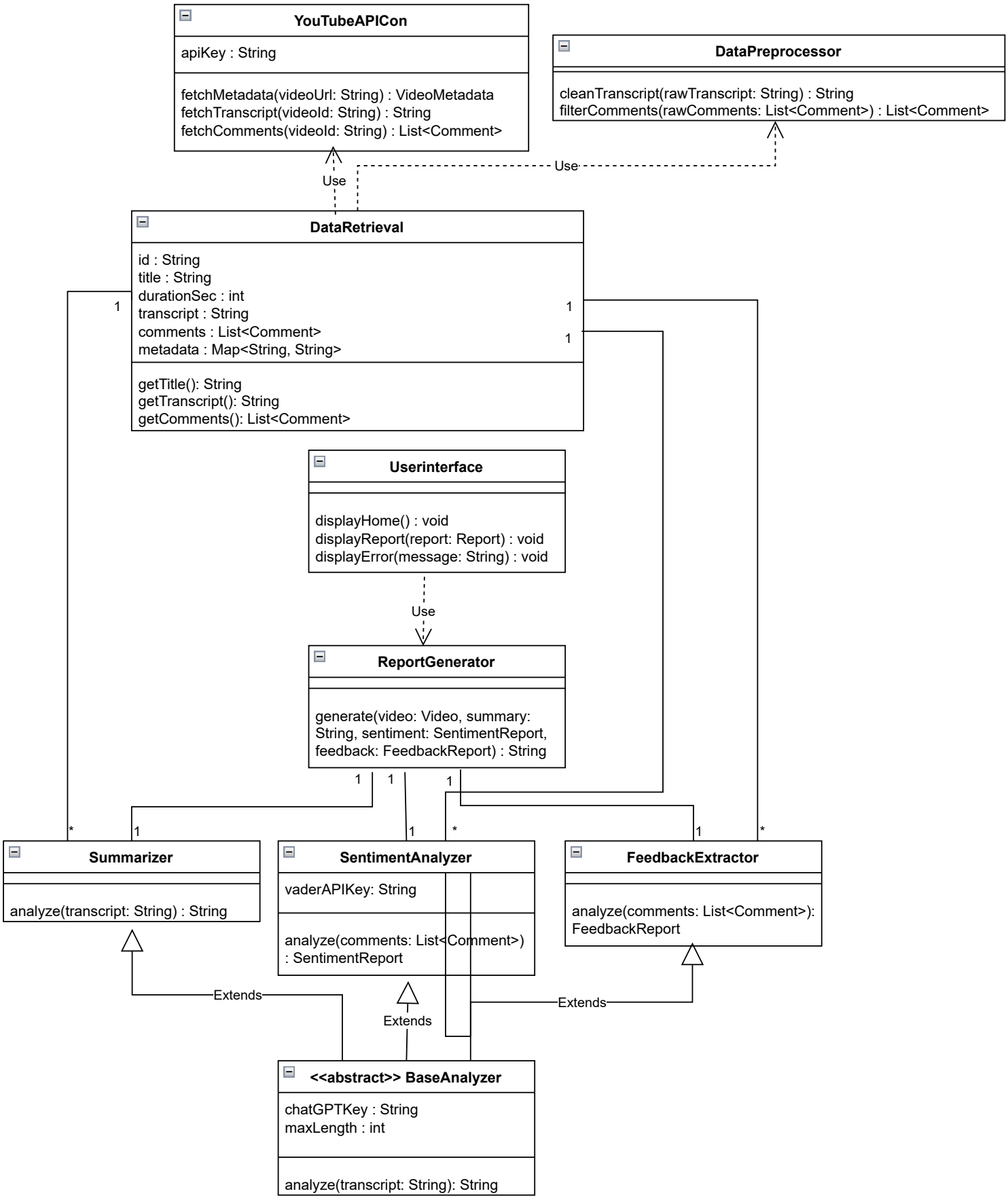
Use Case Name	Display Sentiment Analysis Results
Reference to Product Backlog	PB-22
Scope	YouTube Video Analysis System
Level	User-goal
Primary Actor	User
Stakeholders and Their Interests	User: Expects to see clear sentiment classifications (positive, negative, neutral) for comments System: Must present sentiment data in an understandable manner
Preconditions	Sentiment analysis results from UC-16 are available network connection is stable
Success Criteria	Sentiment analysis results are accurately displayed on the results page
Main Success Scenario	1. Web Server retrieves sentiment analysis results 2. Formats the results with clear labels for positive, negative, and neutral 3. Displays the results on the results page
Extensions	If sentiment results are missing or unclear, display an error message
Special Requirements	Results must clearly indicate each comment's sentiment category
Technology and Data Variations	Standard web display format
Frequency	Every time comment sentiment is processed
Open Issues	None

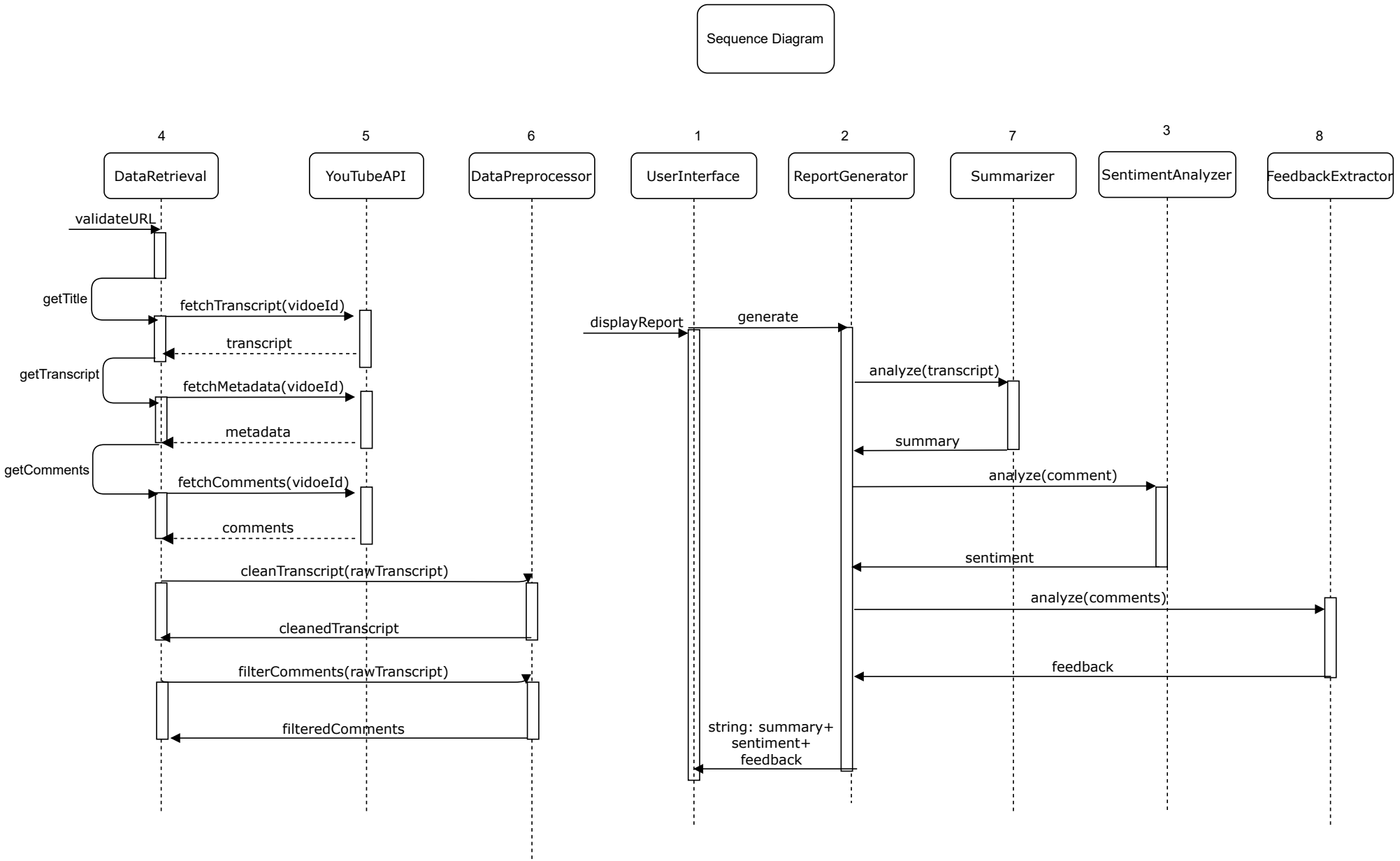


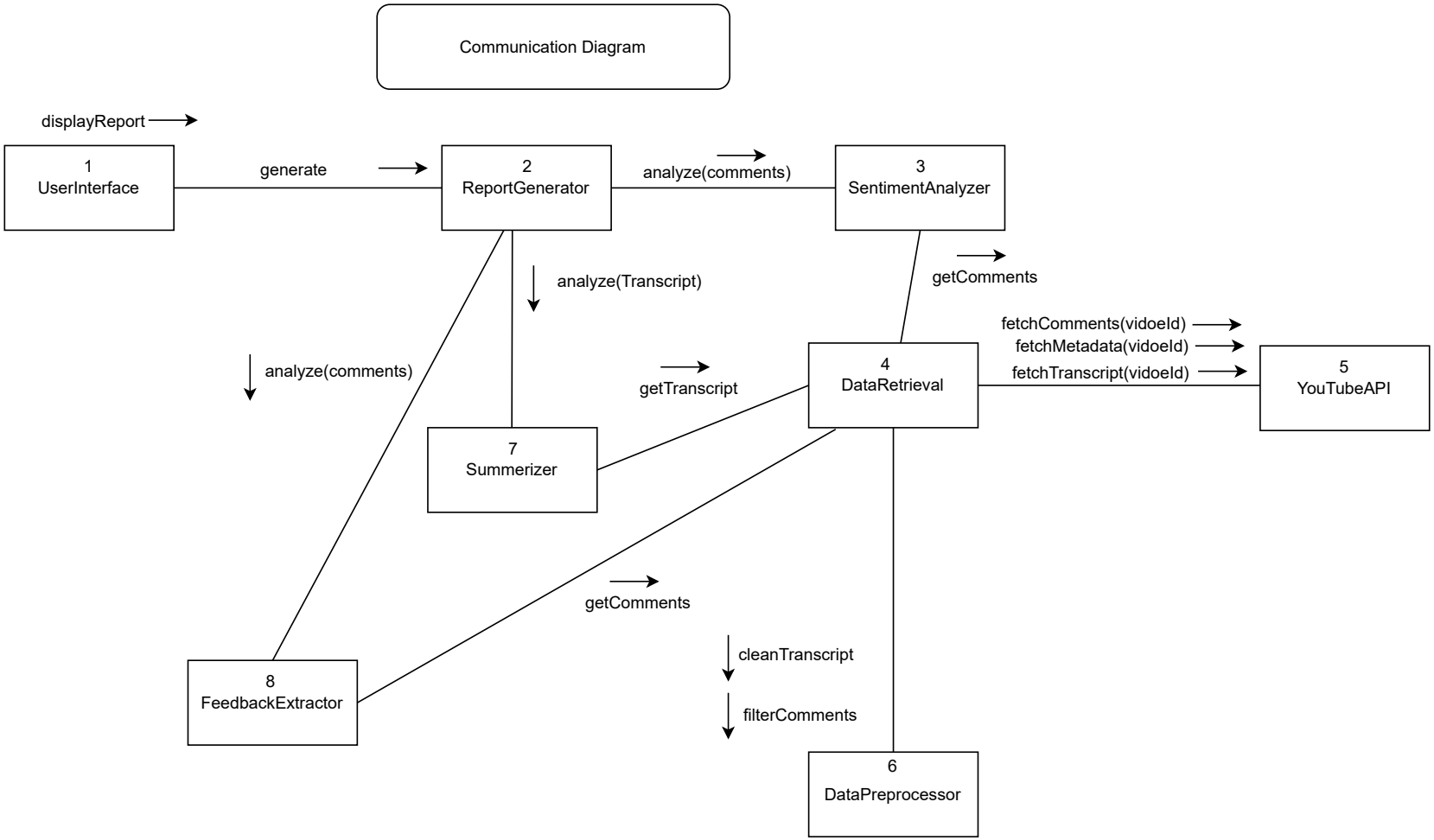
Use Case: Display Structured Feedback Report

Use Case Name	Display Structured Feedback Report
Reference to Product Backlog	PB-23
Scope	YouTube Video Analysis System
Level	User-goal
Primary Actor	User
Stakeholders and Their Interests	User: Expects a detailed report that clearly separates positive feedback from improvement suggestions System: Must generate and display the report in an organized manner
Preconditions	Feedback from UC-18 and UC-19 is available network connection is stable
Success Criteria	The feedback report is displayed with two distinct sections labeled “What’s Working” and “Needs Improvement”
Main Success Scenario	1. Web Server retrieves the positive and improvement feedback 2. Organizes the feedback into two distinct sections 3. Displays the structured report on the results page
Extensions	If the report cannot be generated, display an error message
Special Requirements	Report must clearly delineate the two types of feedback
Technology and Data Variations	Standard web display format
Frequency	Every time a video analysis is completed
Open Issues	None




YouTubeAPICon
apiKey : String
fetchMetadata(videoUrl: String) : VideoMetadata
fetchTranscript(videoId: String) : String
fetchComments(videoId: String) : List<Comment>
DataPreprocessor
cleanTranscript(rawTranscript: String) : String
filterComments(rawComments: List<Comment>) : List<Comment>
Use
Use
DataRetrieval
id : String
title : String
durationSec : int
transcript : String
comments : List<Comment>
metadata : Map<String, String>
getTitle(): String
getTranscript(): String
getComments(): List<Comment>
1
1
1
1
Userinterface
displayHome() : void
displayReport(report: Report) : void
displayError(message: String) : void
Use
ReportGenerator
generate(video: Video, summary: String, sentiment: SentimentReport, feedback: FeedbackReport) : String
1
1
1
*
1
1
*
1
*
Summarizer
analyze(transcript: String) : String
SentimentAnalyzer
vaderAPIKey: String
analyze(comments: List<Comment>) : SentimentReport
FeedbackExtractor
analyze(comments: List<Comment>): FeedbackReport
Extends
Extends
Extends
<<abstract>> BaseAnalyzer
chatGPTKey : String
maxLength : int
analyze(transcript: String): String

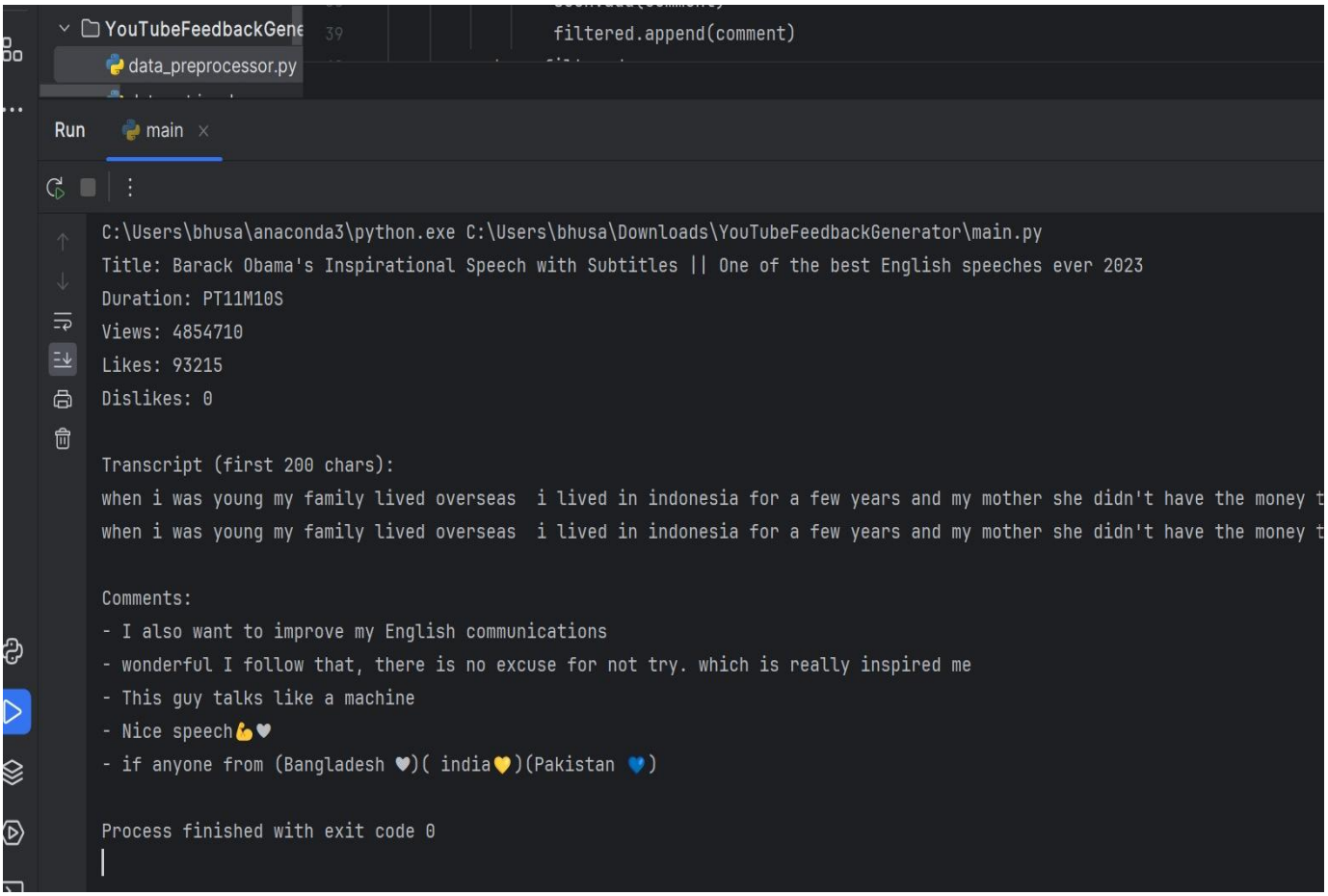

Sequence Diagram
4
5
6
1
2
7
3
8
DataRetrieval
YouTubeAPI
DataPreprocessor
UserInterface
ReportGenerator
Summarizer
SentimentAnalyzer
FeedbackExtractor
validateURL
getTitle
fetchTranscript(vidoeId)
transcript
getTranscript
fetchMetadata(vidoeId)
metadata
getComments
fetchComments(vidoeId)
comments
cleanTranscript(rawTranscript)
cleanedTranscript
filterComments(rawTranscript)
filteredComments
displayReport
generate
analyze(transcript)
summary
analyze(comment)
sentiment
analyze(comments)
feedback
string: summary+
sentiment+
feedback


Communication Diagram
displayReport
1
UserInterface
generate
2
ReportGenerator
analyze(comments)
3
SentimentAnalyzer
analyze(Transcript)
getComments
fetchComments(vidoeId)
fetchMetadata(vidoeId)
fetchTranscript(vidoeId)
analyze(comments)
getTranscript
4
DataRetrieval
5
YouTubeAPI
7
Summerizer
getComments
cleanTranscript
filterComments
8
FeedbackExtractor
6
DataPreprocessor

Implementation of Sprint 1:

- **Extract user feedback from comments using API**
- **URL validation and meta data retrieval:**
- **Transcript and comment retrieval**: fetch video transcript, retrieve at least 100 comments from APA and exceeds 1 hour.

Output:



The output displays YouTube data retrieved from the YouTube API. The data includes:

i. Metadata: includes data from YouTube videos such as title, duration, views, likes.
ii. Transcript: transcript from YouTube video.
iii. Comments: all comments from YouTube video.

Program Code:

main.py

```python
from data_retrieval import DataRetrieval


1 usage
def main():
    api_key = "AIzaSyBD8EDSacRnoKbCfq9riyk2k2th_LYnZks"
    video_url = "https://www.youtube.com/watch?v=PGUdWfB8nLg"

    dr = DataRetrieval(api_key, video_url)

    try:
        dr.validate_url()
        metadata = dr.get_metadata()
        transcript = dr.get_transcript()
        comments = dr.get_comments()

        print(f"Title: {metadata['title']}")
        print(f"Duration: {metadata['duration']}")
        print(f"Views: {metadata['views']}")
        print(f"Likes: {metadata['likes']}")
        print(f"Dislikes: {metadata['dislikes']}")

        print("\nTranscript (first 200 chars):")
        print(transcript[:200])
        print(transcript)

        print("\nComments:")
        for comment in comments[:5]:
            print(f"- {comment}")

    except ValueError as e:
        print(f"Error: {e}")
    except ConnectionError as e:
        print(f"Connection error: {e}")
    except Exception as e:
        print(f"Unexpected error: {e}")


if __name__ == "__main__":
    main()
```

data_preprocessor.py

main.py data_preprocessor.py

```python
import re
from typing import List


4 usages
class DataPreprocessor:
    1 usage
    @staticmethod
    def remove_filler_words(transcript: str) -> str:
        """Removes common filler words from the transcript"""
        fillers = ["um", "uh", "like", "you know", "so", "actually", "basically", "literally", "right"]
        pattern = r'\b(' + '|'.join(fillers) + r')\b'
        cleaned_transcript = re.sub(pattern, repl: '', transcript, flags=re.IGNORECASE)
        return cleaned_transcript

    1 usage
    @staticmethod
    def clean_transcript(raw_transcript: str) -> str:
        """Cleans transcript by removing timestamps, filler words, and spam content"""
        cleaned = re.sub( pattern: r'\[\d+:\d+(?::\d+)?]', repl: '', raw_transcript)  # Remove timestamps
        cleaned = DataPreprocessor.remove_filler_words(cleaned)  # Remove filler words
        return cleaned.lower()

    1 usage
    @staticmethod
    def filter_comments(raw_comments: List[str]) -> List[str]:
        # Remove duplicate comments
        seen = set()
        filtered = []
        # Remove spam content using regex patterns
        spam_patterns = [
            r'\b(free|subscribe|click here|visit our website|spam)\b',
            r'(http|https)://\S+',  # URLs
            r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b',  # Emails
            r'\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b'  # IP addresses
        ]
        for comment in raw_comments:
            if comment and comment not in seen:
                for pattern in spam_patterns:
                    comment = re.sub(pattern, repl: '', comment, flags=re.IGNORECASE)
                seen.add(comment)
                filtered.append(comment)
        return filtered

```

data_retrieval.py

main.py | data_preprocessor.py | data_retrieval.py | youtubeAPICon.py

```python
from youtubeAPICon import YouTubeAPICon
from data_preprocessor import DataPreprocessor
import re


2 usages
class DataRetrieval:
    def __init__(self, api_key: str, video_url: str):
        self.api = YouTubeAPICon(api_key)
        self.video_url = video_url
        self.video_id = None

    1 usage
    def validate_url(self):
        """Validate YouTube URL and extract video ID"""
        pattern = r"^https?://(www\.)?(youtube\.com|youtu\.be)/(watch\?v=)?([a-zA-Z0-9_-]{11})"
        match = re.match(pattern, self.video_url)
        if not match:
            raise ValueError("Invalid YouTube URL")
        self.video_id = match.group(4)

    1 usage
    def get_metadata(self) -> dict:
        """Fetch and return video metadata"""
        if not self.video_id:
            raise RuntimeError("URL validation required before data retrieval")
        metadata = self.api.fetch_metadata(self.video_id)
        # Enforce operational constraints
        if metadata["duration"].startswith("PT") and "H" in metadata["duration"]:  # Checks for 1+ hour videos
            raise ValueError("Video exceeds the allowed duration of 1 hour")

        return metadata
```

```python
    2 usages (1 dynamic)
    def get_transcript(self) -> str:
        """Fetch and return cleaned transcript"""
        if not self.video_id:
            raise RuntimeError("URL validation required before data retrieval")
        raw_transcript = self.api.fetch_transcript(self.video_id)
        return DataPreprocessor.clean_transcript(raw_transcript)

    1 usage
    def get_comments(self) -> list[str]:
        """Fetch and return filtered comments"""
        if not self.video_id:
            raise RuntimeError("URL validation required before data retrieval")
        raw_comments = self.api.fetch_comments(self.video_id)

        if len(raw_comments) < 100:
            raise ValueError("Video does not have the required minimum of 100 comments")

        return DataPreprocessor.filter_comments(raw_comments)
```

youtubeAPICon.py

main.py | data_preprocessor.py | data_retrieval.py | youtubeAPICon.py

```python
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
from youtube_transcript_api import YouTubeTranscriptApi
from typing import Dict, List


2 usages
class YouTubeAPICon:
    def __init__(self, api_key: str):
        self.api_key = api_key
        self.youtube = build('youtube', 'v3', developerKey=self.api_key)

    1 usage
    def fetch_metadata(self, video_id: str) -> Dict[str, str]:
        try:
            response = self.youtube.videos().list(
                part="snippet,contentDetails,statistics",
                id=video_id
            ).execute()
            items = response.get('items', [])
            if not items:
                raise ValueError("Video not found")

            metadata = items[0]
            return {
                "title": metadata['snippet']['title'],
                "duration": metadata['contentDetails']['duration'],
                "views": metadata['statistics']['viewCount'],
                "likes": metadata['statistics'].get('likeCount', '0'),
                "dislikes": metadata['statistics'].get('dislikeCount', '0')
            }
        except HttpError as e:
            raise ConnectionError(f"Metadata API error: {e.resp.status}") from e

    1 usage
```

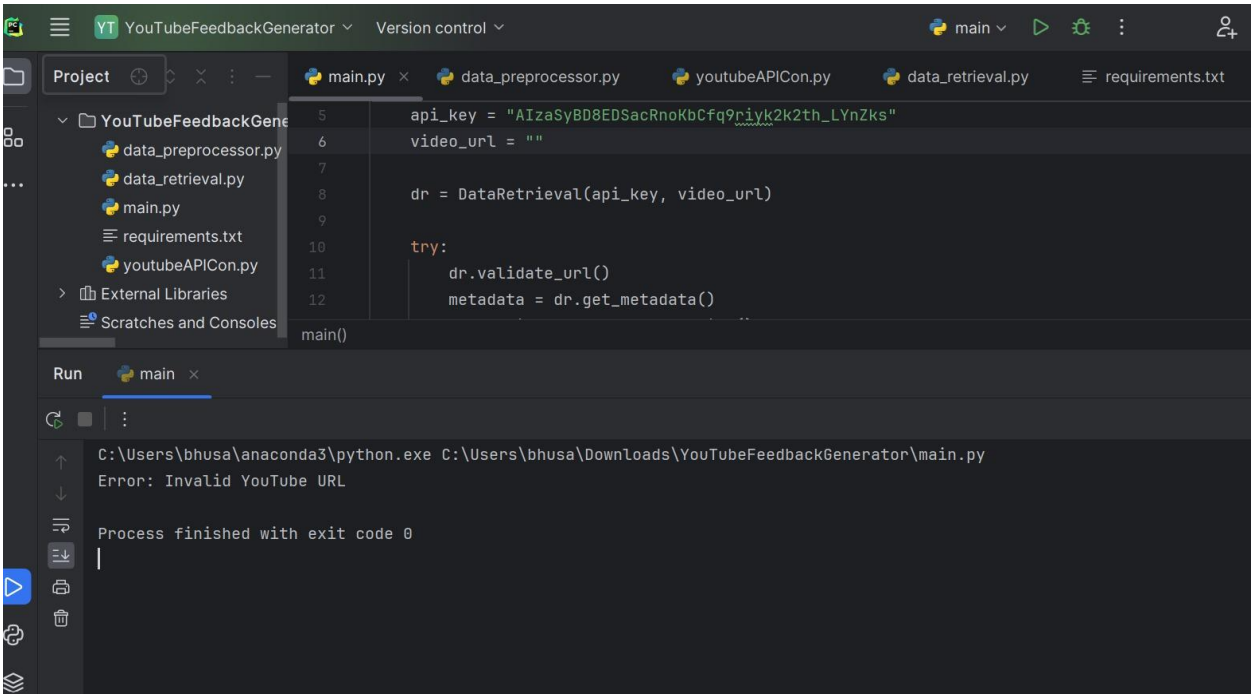
main.py | data_preprocessor.py | data_retrieval.py | youtubeAPICon.py

```python
    def fetch_comments(self, video_id: str, max_comments: int = 3000) -> List[str]:
        """Fetch up to max_comments (default 3000) top-level comments from YouTube API"""
        comments = []
        try:
            request = self.youtube.commentThreads().list(
                part="snippet",
                videoId=video_id,
                textFormat="plainText",
                maxResults=100  # Fetch 100 comments per page (max allowed by YouTube API)
            )
            while request and len(comments) < max_comments:
                response = request.execute()
                for item in response['items']:
                    comments.append(
                        item['snippet']['topLevelComment']['snippet']['textDisplay']
                    )
                    if len(comments) >= max_comments:
                        break  # Stop fetching more comments once limit is reached

                # Check if there are more pages of comments
                if 'nextPageToken' in response and len(comments) < max_comments:
                    request = self.youtube.commentThreads().list(
                        part="snippet",
                        videoId=video_id,
                        textFormat="plainText",
                        maxResults=100,
                        pageToken=response['nextPageToken']
                    )
                else:
                    break  # No more pages, exit loop

            return comments
        except HttpError as e:
            raise ConnectionError(f"Comments API error: {e.resp.status}") from e

    1 usage
    def fetch_transcript(self, video_id: str) -> str:
        try:
            transcript_list = YouTubeTranscriptApi.get_transcript(video_id)
            transcript = ' '.join([entry['text'] for entry in transcript_list])
            return transcript
        except Exception as e:
            return f"Transcript unavailable: {str(e)}"
```

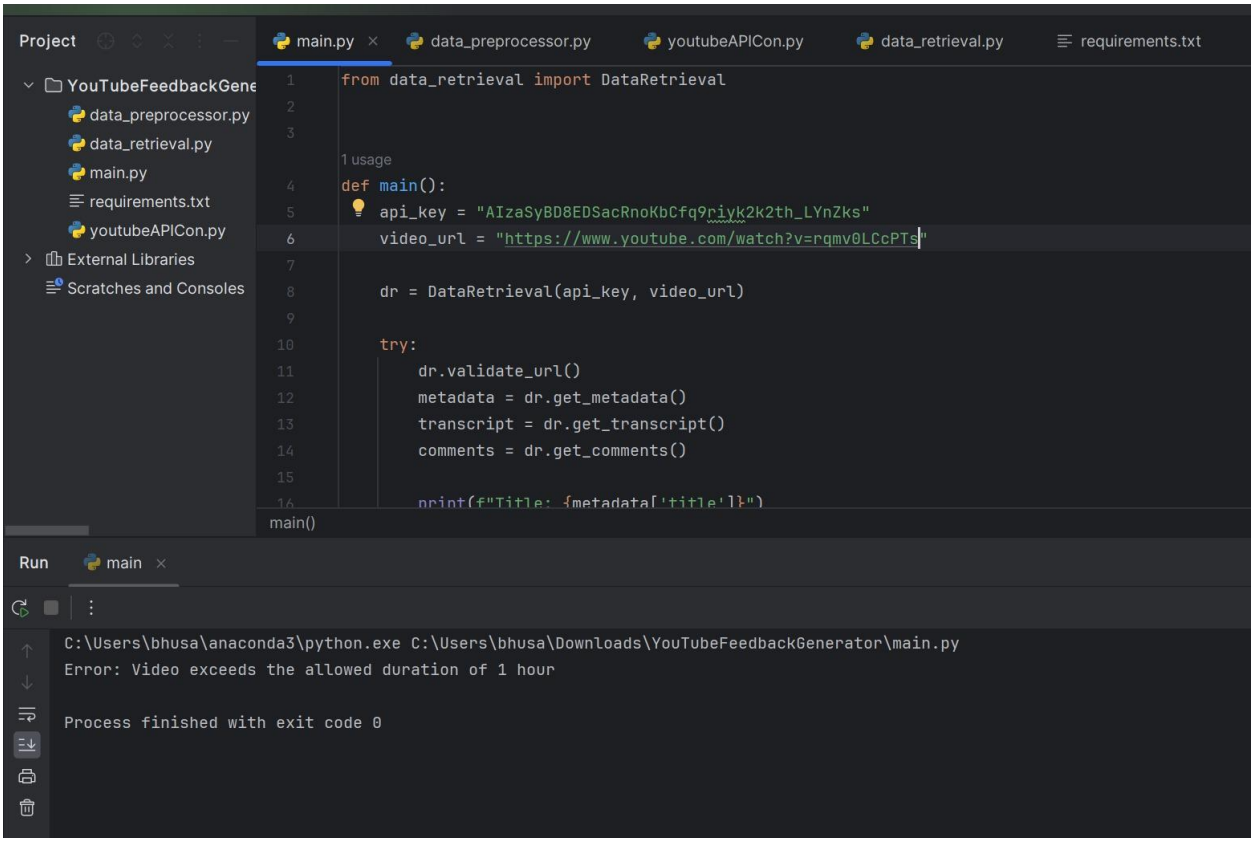
YouTubeAPICon › fetch_comments()

Test Cases:

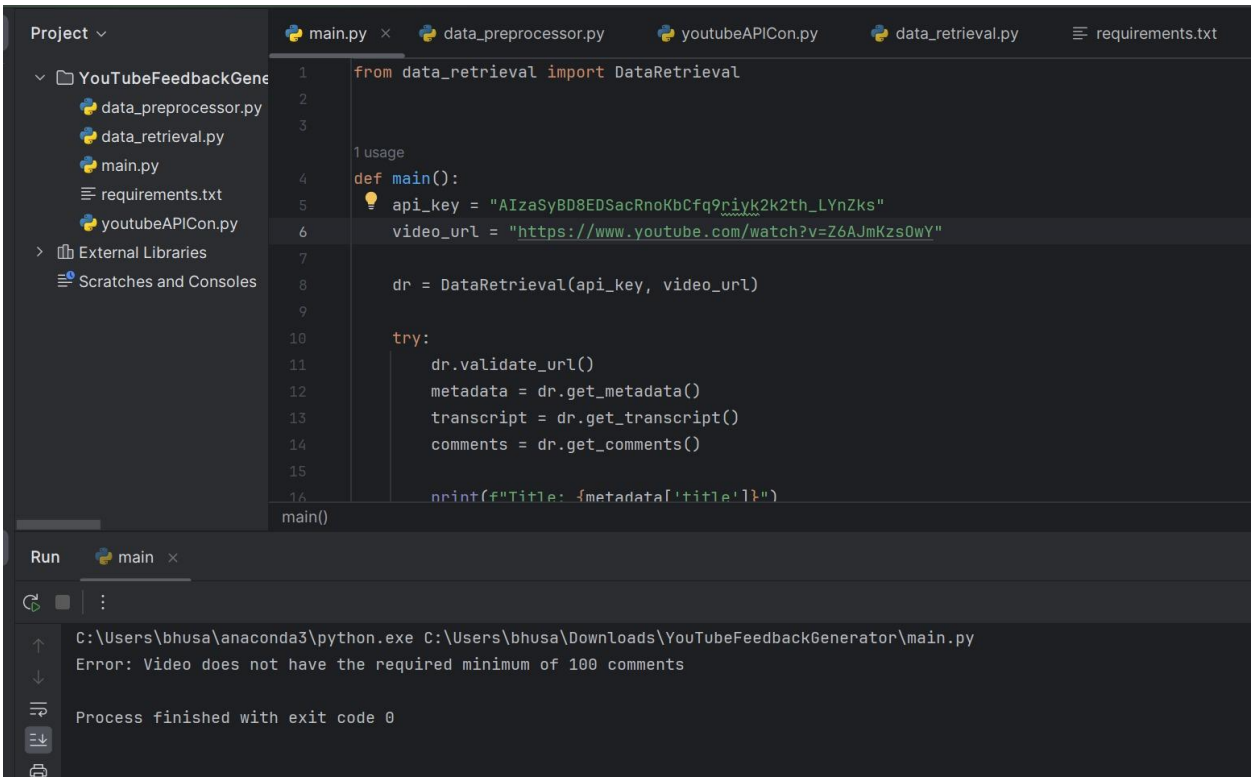
Test Case 1: When the user searches without any YouTube URL or the URL entered is invalid.



Test Case 2: When YouTube video duration is more than 1 hour.



Test Case 3: When YouTube video has less than 100 comments. For this instance, the video has 0 comments.



Reflection of Sprint 1:

What worked well:

- YouTube URL validation and video ID extraction are correctly implemented in validate_url() in data_retrieval.py.
- Metadata retrieval (title, description, duration) works well, enforcing constraints to exclude videos longer than 1 hour.
- Transcript retrieval and cleaning successfully remove timestamps and filler words using clean_transcript() in data_preprocessor.py.
- Comment filtering partially works; duplicates and some spam patterns are removed.

What didn't work well:

- **Filtering Comments Is Complex:**
  Spam detection is challenging since spam formats vary.
  Simple regex filters might not be enough—NLP-based techniques should be explored.
  Irrelevant comments require topic modeling to determine whether they are related to the video.
- **Comment processing speed:**
  A 3000-comment limit was set to balance performance, but if scaling up, processing time could increase significantly.
  Batch processing or asynchronous API calls may be needed for efficiency.

Lessons learnt:

- Underestimated the complexity of cleaning the data including comments and transcripts.

Impact on project:

- Could have worked on less User Stories to complete all user stories deliverables promised in Sprint 1.

Invoice (in hours):

Documentation	Ayush – 32, Anthony – 32, Nabin – 32
Research	Ayush – 32, Anthony – 32, Nabin – 34
Design	Ayush – 16, Anthony – 17, Nabin – 16
Code	Ayush – 14, Anthony – 12, Nabin – 12
Testing	Ayush – 8, Anthony – 6, Nabin – 6

Estimated for Remaining Work: As per Product Backlog 50-20 = 30 days. 5 extra days to complete what didn't work well in Sprint 1.
Total: 35 days

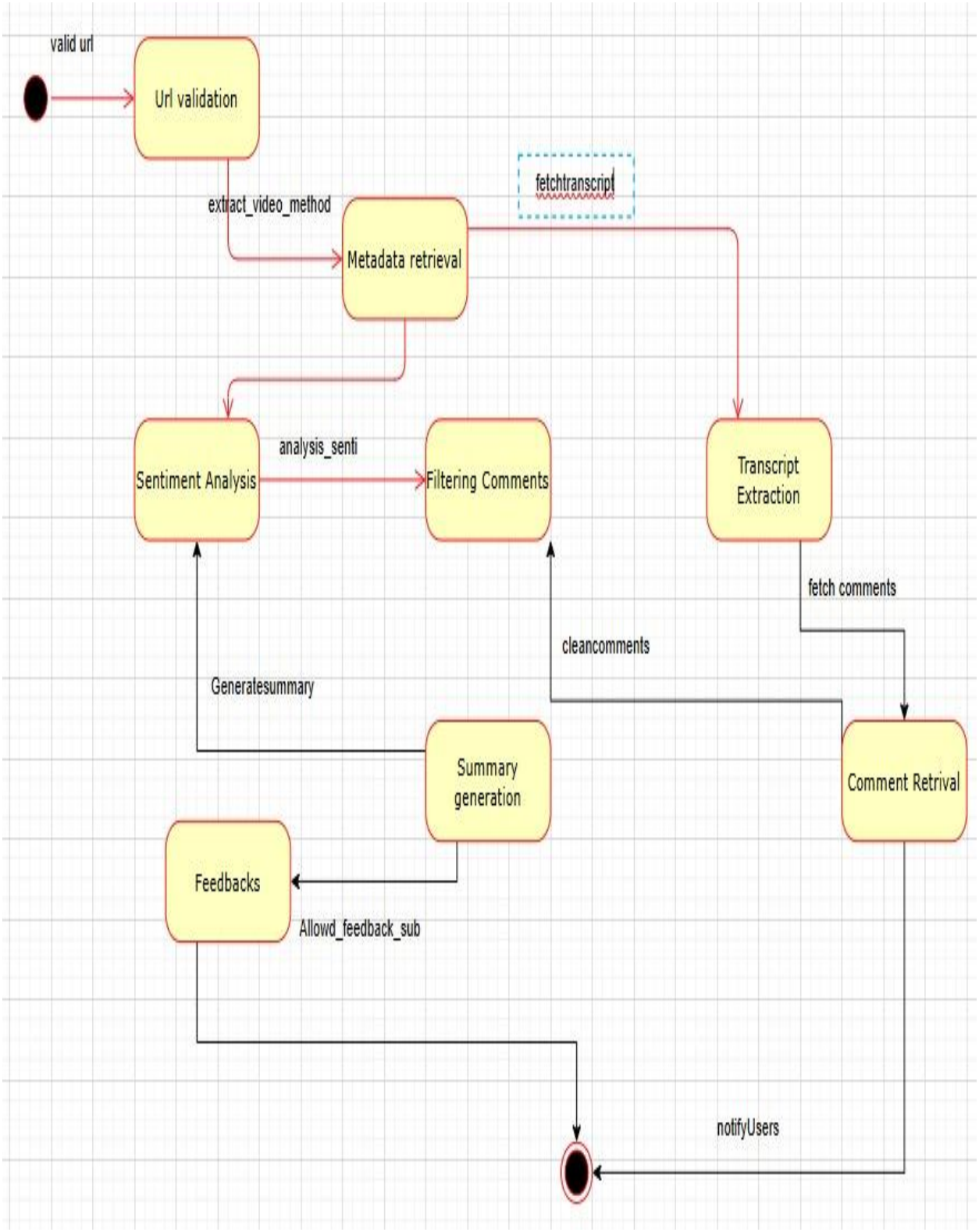
Product Backlogs for Sprint 2 & 3:

	ID	Product Backlog	Estimation	Priority
Sprint 2	14	The Web Server can send the clean transcript to the ChatGPT API to generate a concise video summary in paragraph form and save it.	5	14
	15	Define and implement the interface for the VADER sentiment analysis tool.	2	15
	16	The Web Server will use VADER to classify filtered comments as positive, negative, or neutral, and store the sentiment analysis results.	3	16
	17	Define and implement the interface with the ChatGPT API for processing filtered comments.	1	17
Sprint 3	18	Extract actionable "What's Working" feedback from filtered comments using the ChatGPT API.	1	18
	19	Extract actionable "Needs Improvement" feedback from filtered comments using the ChatGPT API.	1	19
	20	Generate a structured feedback report categorizing "What's Working" and "Needs Improvement" and store the report.	2	20

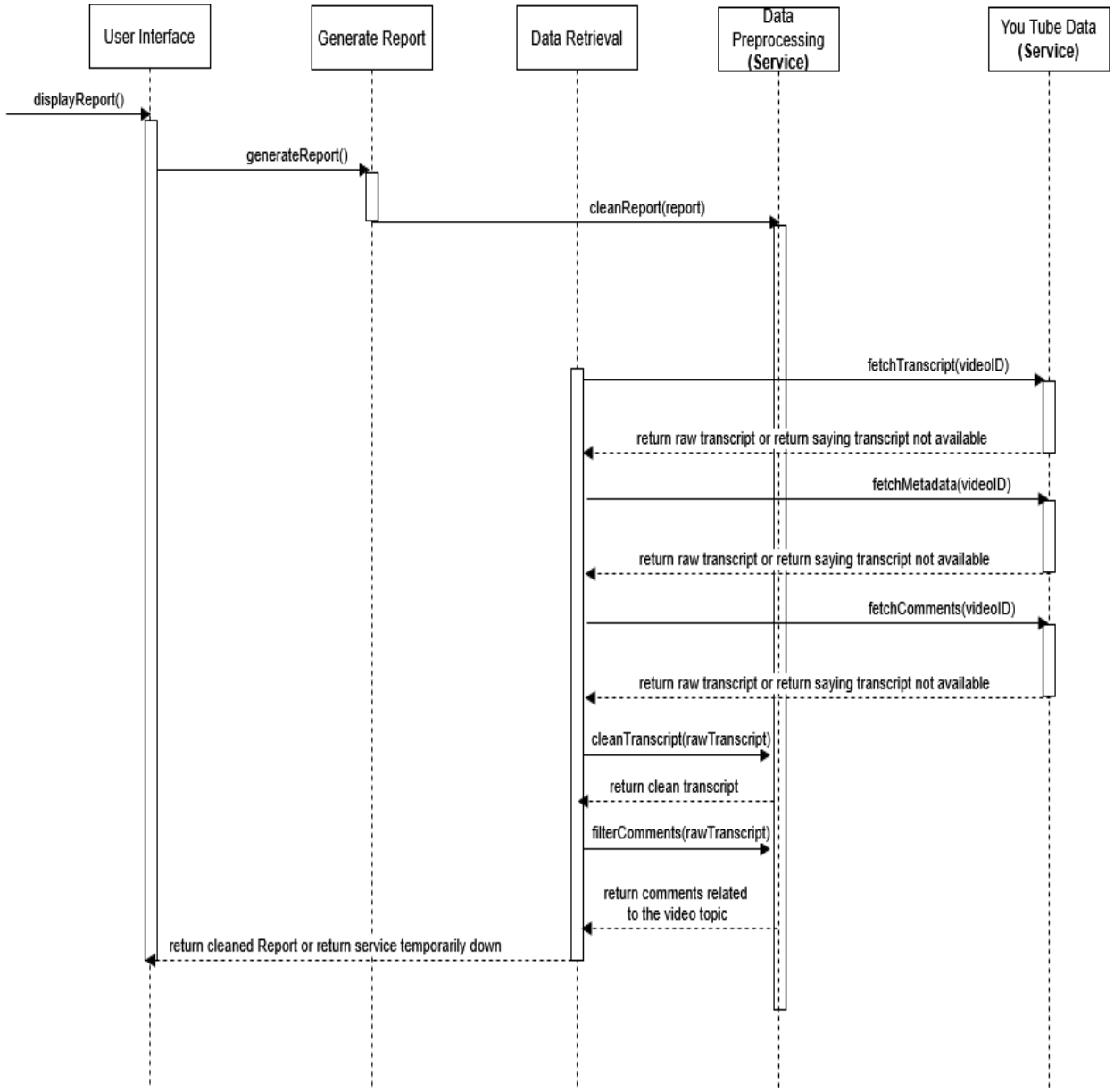
Tools Used:

- **Software used for this system** – PyCharm
- **Software used to produce your documentation** – Microsoft Word and PowerPoint
- **Other Tools** – https://www.draw.io/

State diagram



SOA SERVICES



Implementation of Sprint 2:

- Transcript summarization completed
- Sentiment analysis integrated
- Feedback extraction implemented
- Basic UI added for displaying results

Output:

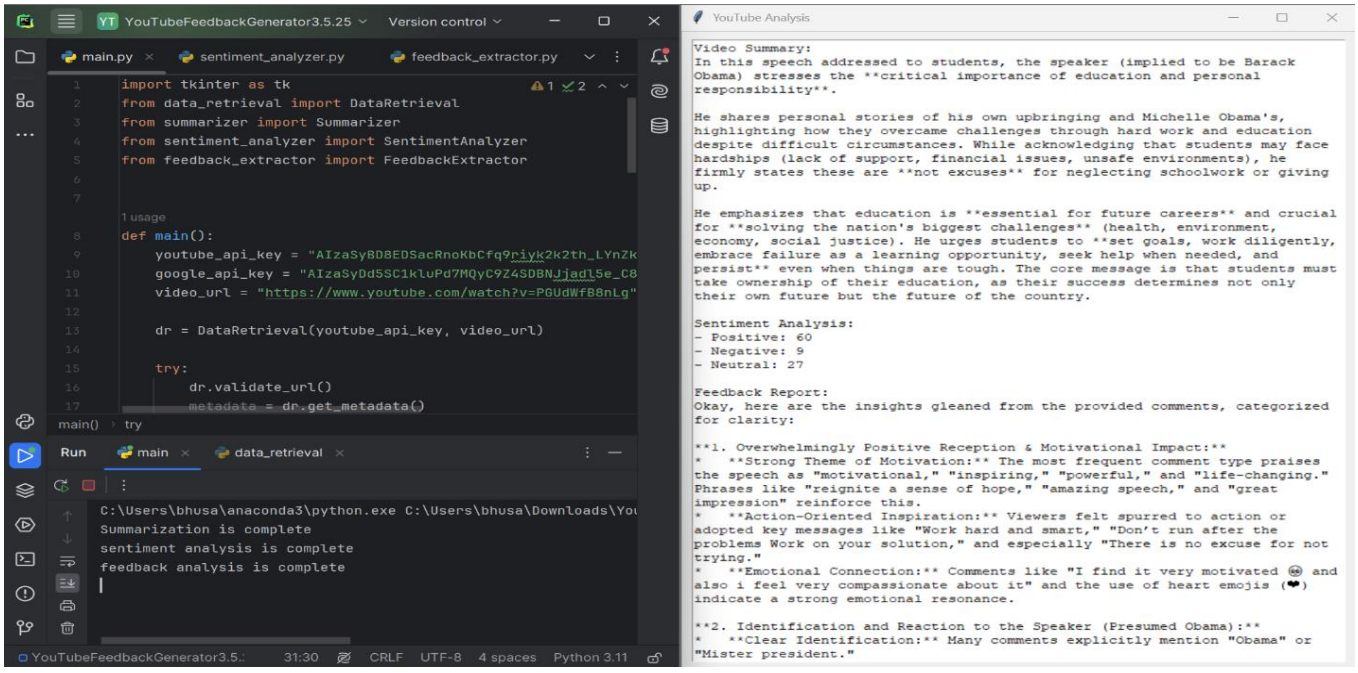
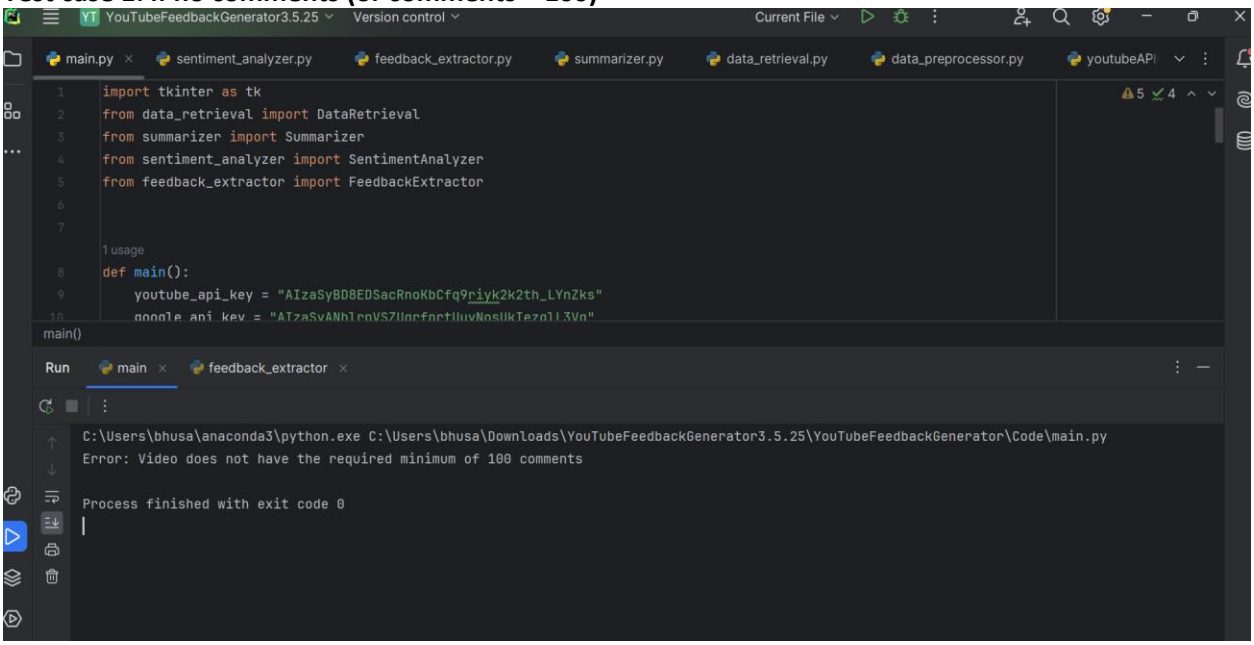


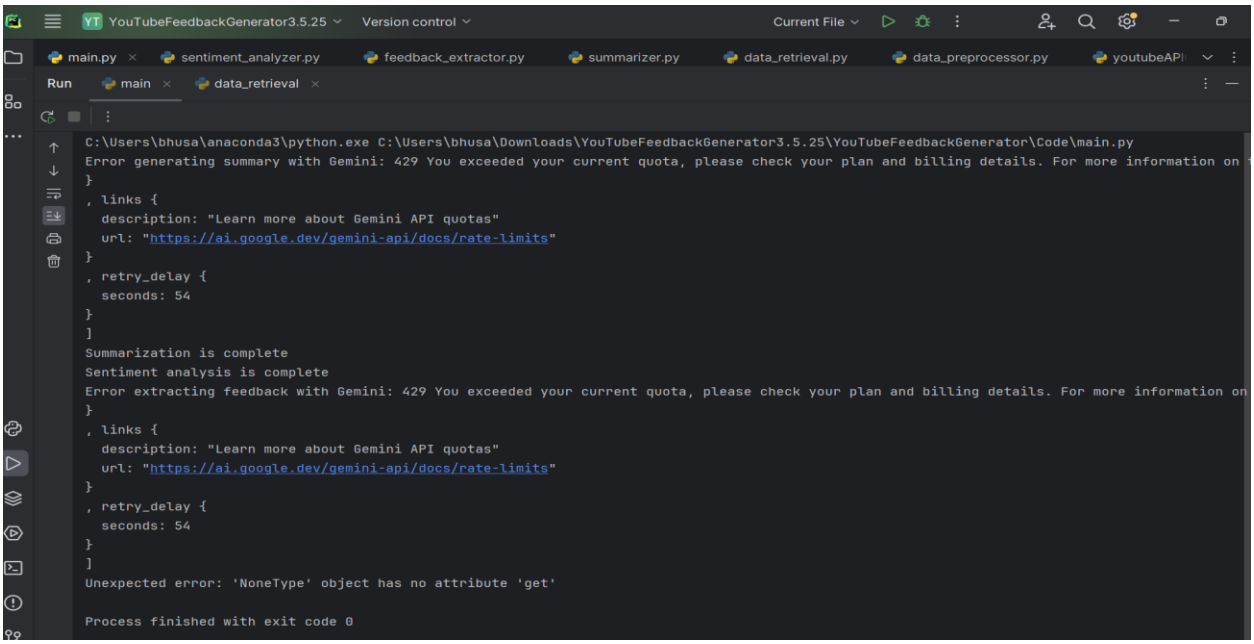
Fig: Normal Demo of project

Test Cases

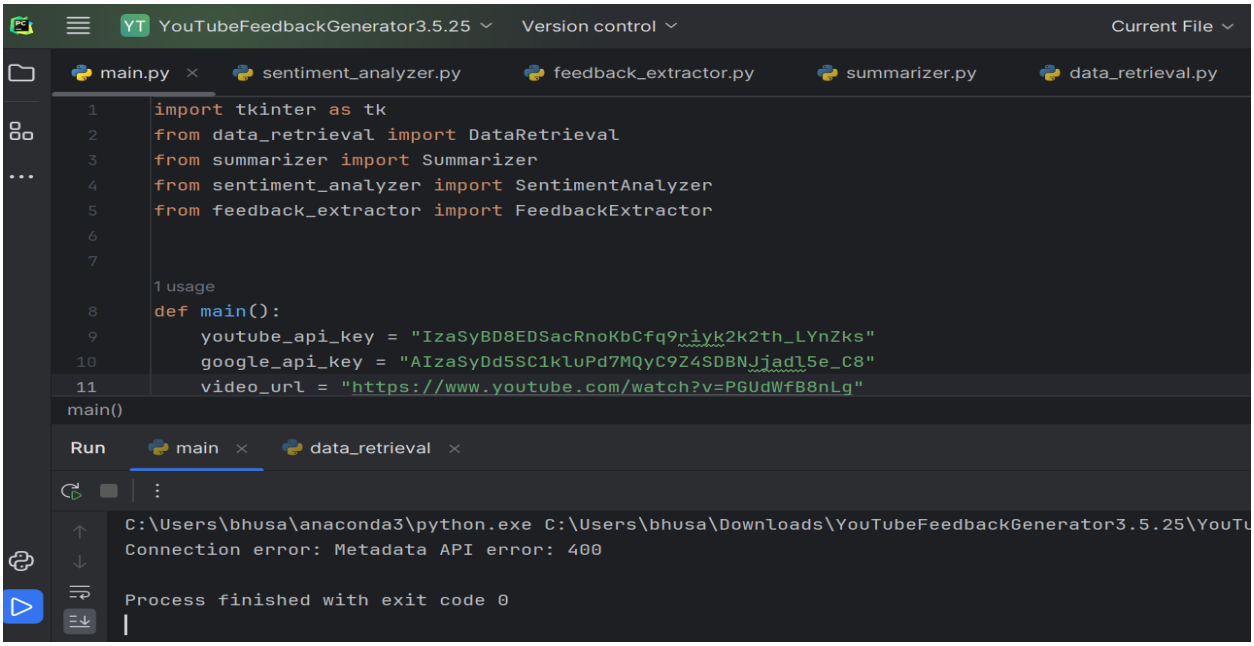
Test case 1: if no comments (or comments < 100)



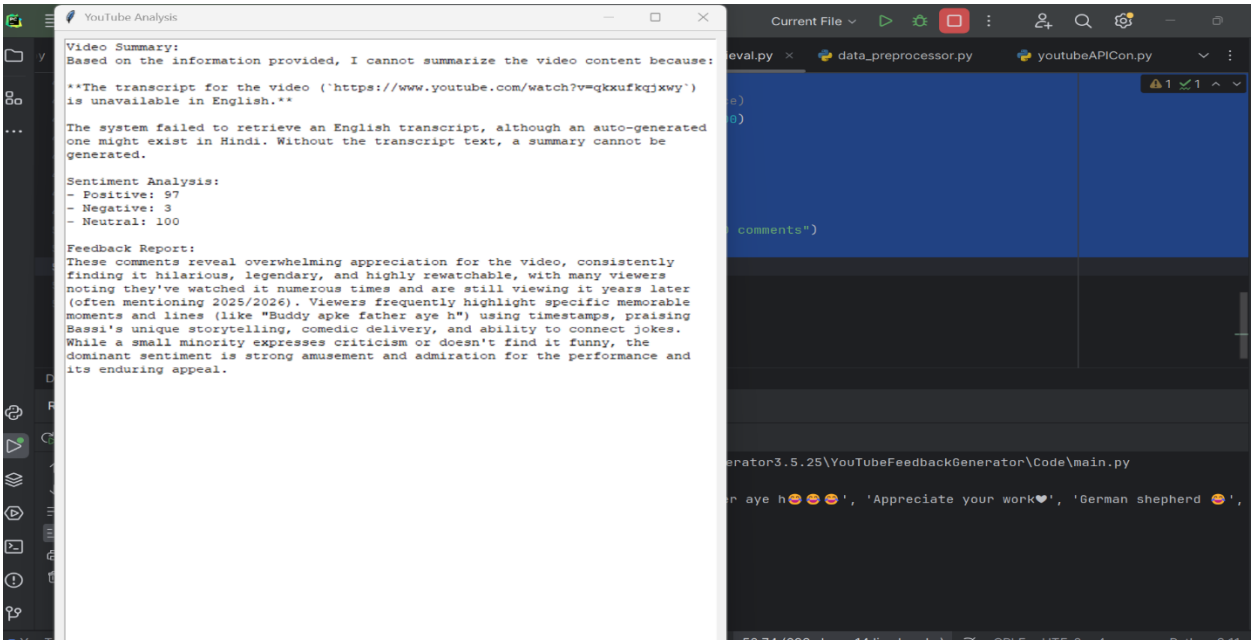
Test case 2: If the API limit exceeded



Test case 3: If API not connected



Test case 4: IF no transcript in video



Program Code:

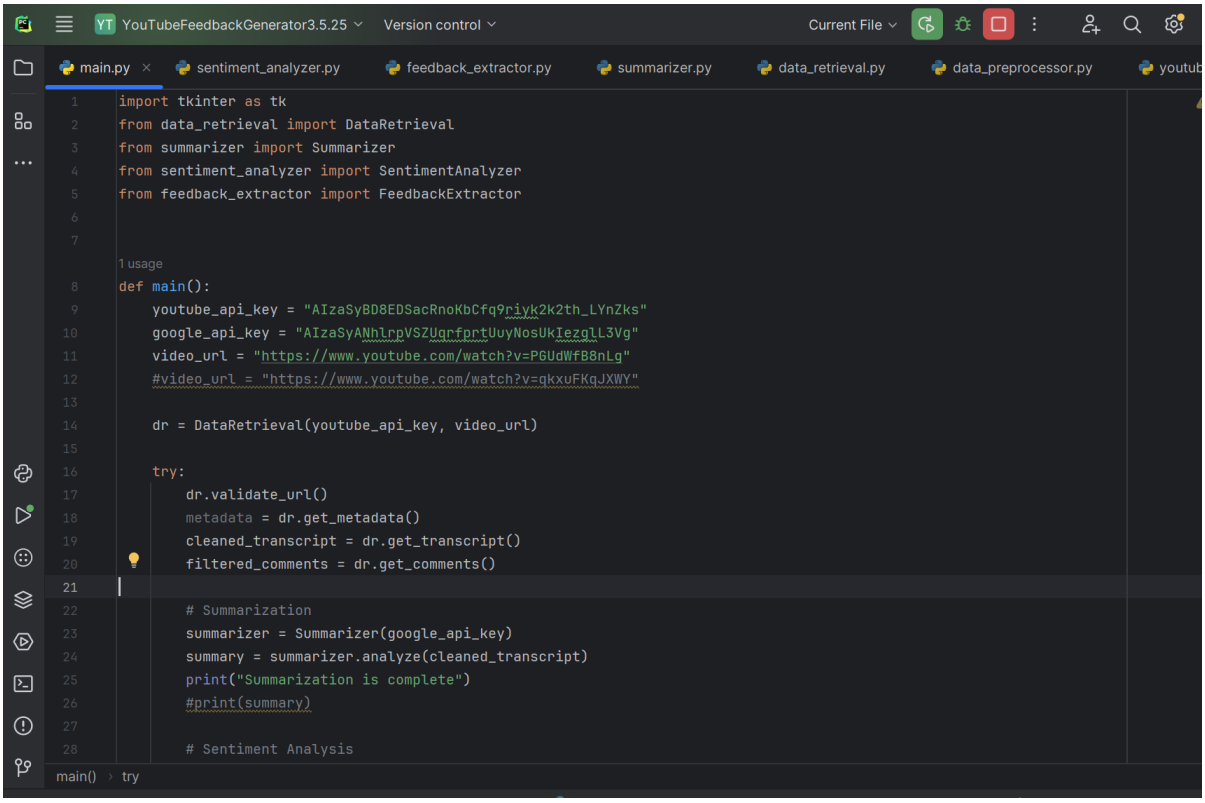


Fig: Main code 1

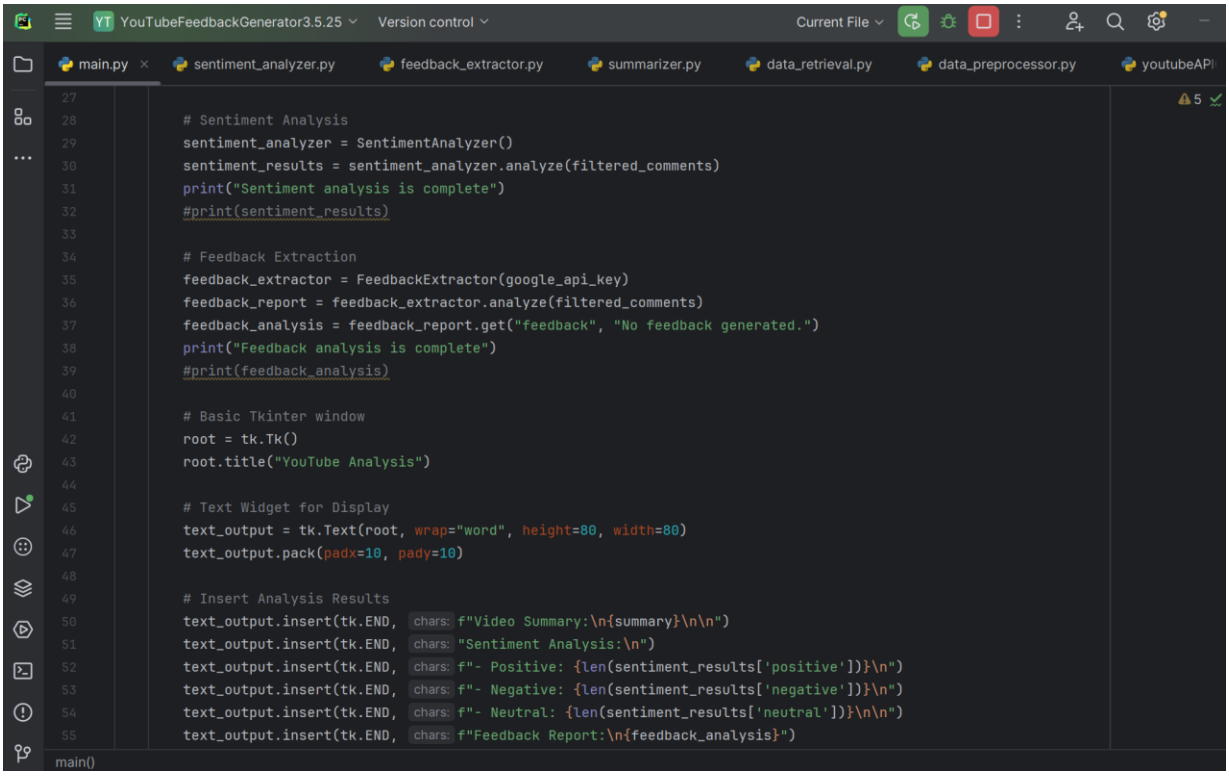


Fig: Main code 2

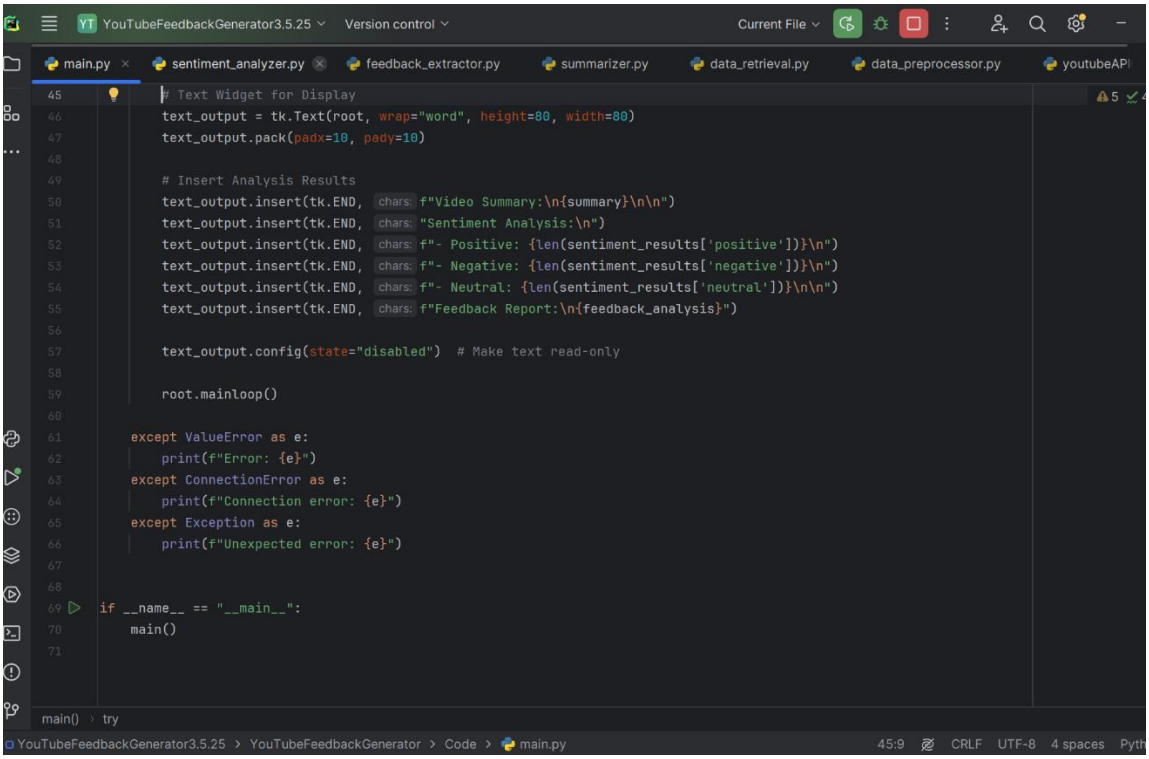


Fig: Main code 3

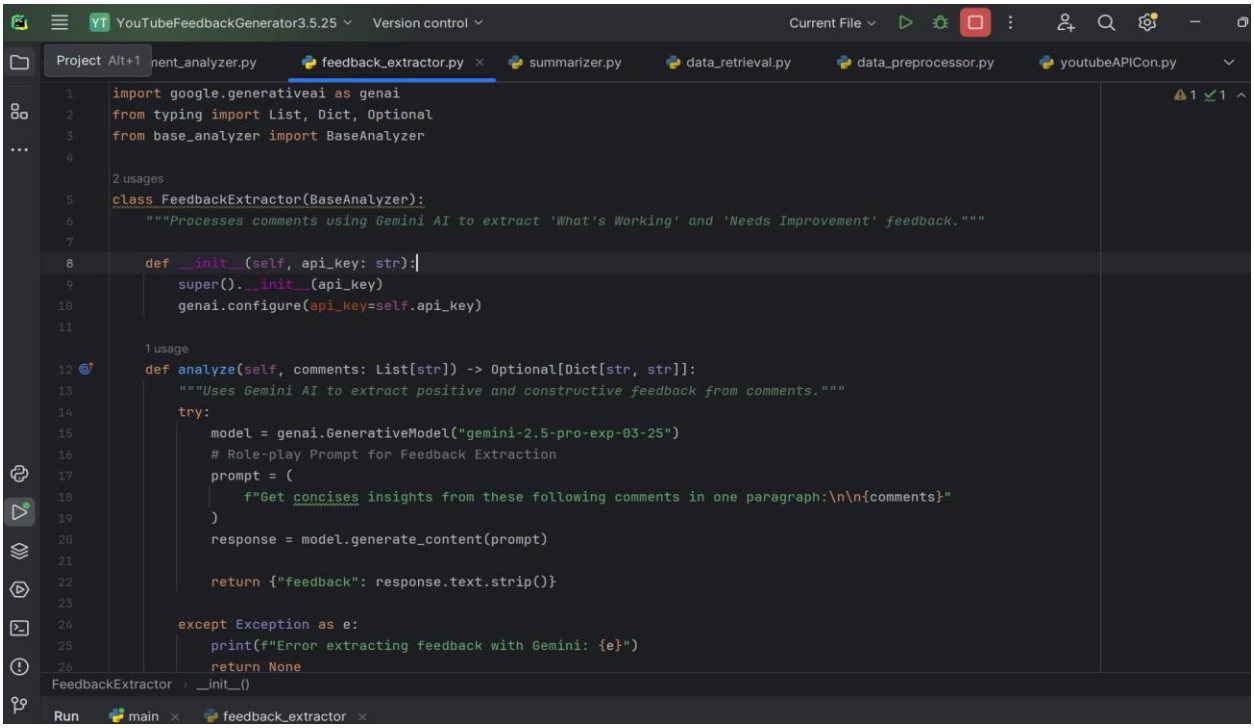


Fig: Feedback code

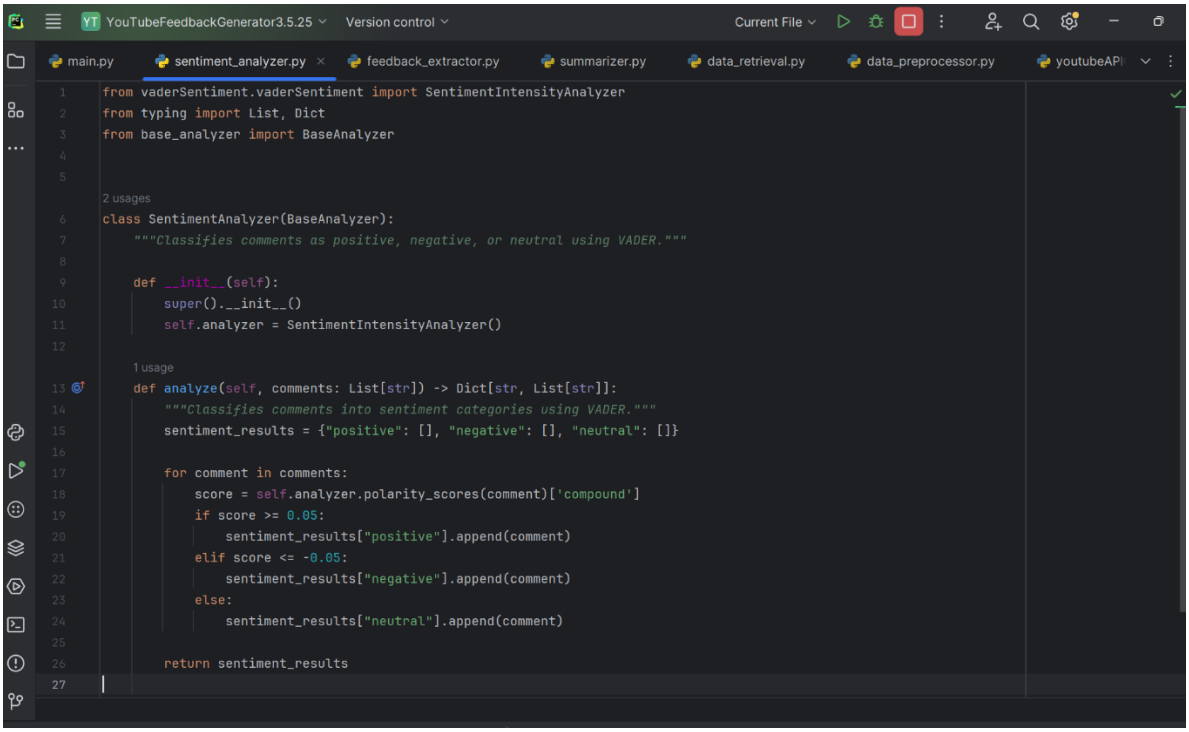


Fig: sentiment analysis code

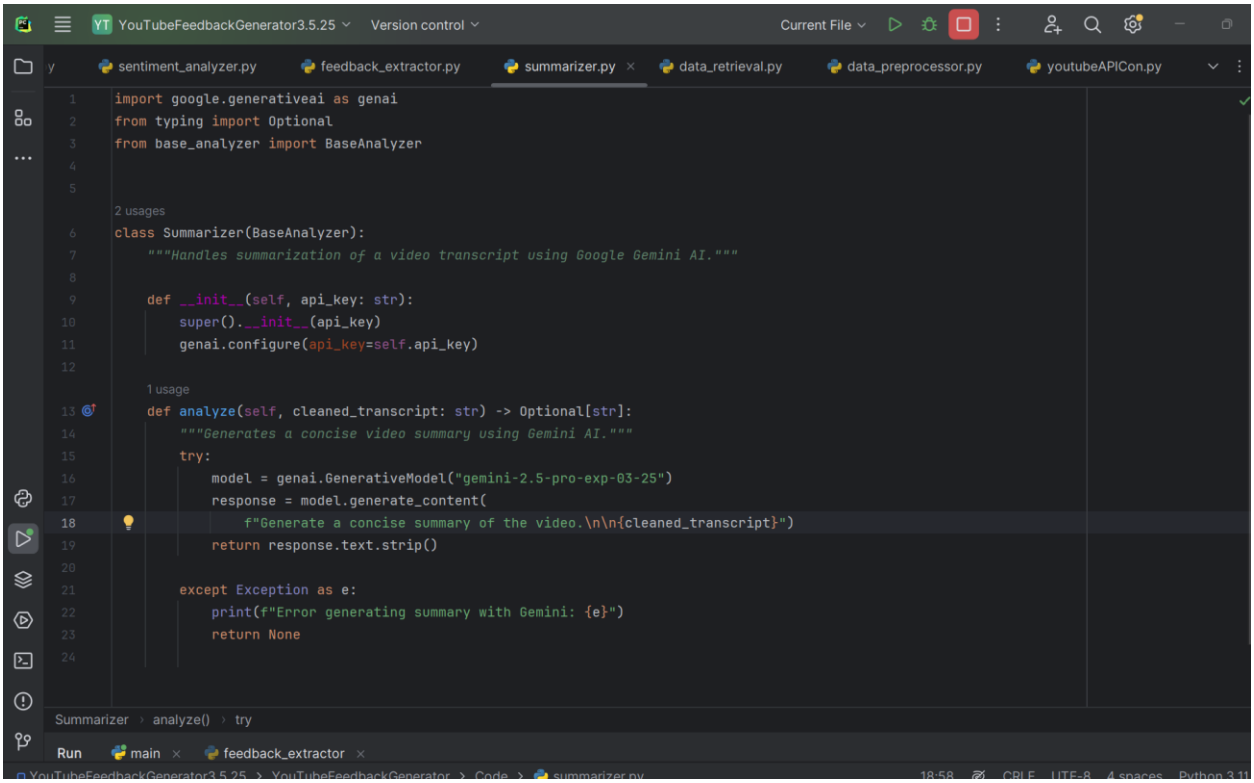


Fig: summarizer code

Reflection of Sprint 2:

What worked well:

1. **Feature Implementation**
    - **Transcript Summarization:** Successfully implemented using Gemini AI to generate concise summaries.
    - **Sentiment Analysis:** Integrated VADER for classifying comments into positive, negative, and neutral categories.
    - **Feedback Extraction:** AI-driven extraction of constructive feedback worked as expected.
    - **Basic UI**: A Tkinter-based interface was added to display the analysis results.
2. **Code Structure & Modularity**
    - The system follows a clear modular design (Summarizer, SentimentAnalyzer, FeedbackExtractor).
    - Separation of concerns makes it easy to maintain and extend.
3. **Error Handling & Stability**
    - Handled potential issues like invalid URLs, API failures, and connection errors gracefully.

Some of the challenges we faced

I. **API Key Security**

- Hardcoded API keys in main.py are a security risk. Moving them to a config file is required for the next sprint.

II. **Processing Efficiency**

- ❖ Handling large comment datasets might slow down execution. Optimization techniques like multithreading need to be researched.

III. UI Improvements

- ❖ The Tkinter interface is functional but minimal. Adding scrollbars, formatting, or interactive elements could enhance usability.

IV. Time Management

- ❖ The team needs to manage the time slot. This sprint was a slight rush.

Lessons learnt:

The lesson learned is to secure API configurations, optimize processing and UI, and improve time management to enhance project execution**.**

Impact on Project

These challenges have led to a strategic reallocation of resources and revised timelines to improve security, efficiency, and overall usability**.**

Invoice (in hours):

Documentation	Ayush – 34, Anthony – 30, Nabin – 32
Research	Ayush – 18, Anthony – 17, Nabin – 16
Design	Ayush – 14, Anthony – 12, Nabin – 15
Code	Ayush – 20, Anthony – 18, Nabin - 16
Testing	Ayush – 8, Anthony – 8, Nabin – 9

Estimated for Remaining Work:
As per Product Backlog: 50 - 20 = 30 days.
5 extra days to complete what didn't work well in Sprint 2.

Total: 35 days

Product Backlog for 3:

ID	Product Backlog	Estimation	Priority
1	As a User, I can enter a YouTube video URL on the home page and click "Search" to initiate video data retrieval (metadata, transcript, and comments).	3	1
18	Extract actionable "What's Working" feedback from filtered comments using the ChatGPT API.	1	18
19	Extract actionable "Needs Improvement" feedback from filtered comments using the ChatGPT API.	1	19
20	Generate a structured feedback report categorizing "What's Working" and "Needs Improvement" and store the report.	2	20
21	As a User, I can view a results page that displays the compiled video metadata (e.g., title, description, duration).	3	21
22	As a User, I can view a results page that displays the generated video summary.	2	22
23	As a User, I can view a results page that displays detailed sentiment analysis results of comments.	2	23
24	As a User, I can view a results page that displays the structured feedback report categorizing "What's Working" and "Needs Improvement".	3	24

Tools Used:

- **Software used for this system – PyCharm**
- **Software used to produce your documentation – Microsoft Word and PowerPoint**
- **Other Tools – https://www.draw.io/**

1. Quality Attributes – Runtime Qualities

Attribute	Bus Rank	Design align	Run time Quality Attribute Documents
Performance			**Definition:** System performance describes attributes of the system including navigational abilities and connectivity with interfacing systems. May include system response time, number of transactions per minute, and other critical performance characteristics which define this project. **Reason for Business Rank <> 3:** Performance is essential to deliver timely video summaries and real-time sentiment analysis for a responsive user experience. **Explanation of Design Alignment <> Meets:** The system employs optimized API calls, efficient data retrieval, and parallel processing to ensure performance targets are met.
Reliability			**Definition:** Reliability defines the ability of the system to operate correctly over time. Includes consistent stability of the system during available hours. **Reason for Business Rank <> 3:** High reliability is critical to maintain user trust and ensure continuous availability of video data and analysis. **Explanation of Design Alignment <> Meets:** Redundant mechanisms, robust error handling, and comprehensive logging are implemented to achieve the required reliability.
Scalability			**Definition:** Scalability defines the parameters of growth within a given period. The solution must scale so that additional capacity can be added expediently in a technically and economically feasible way. **Reason for Business Rank <> 3:** Scalability is needed to handle increasing volumes of video and comment data as the user base grows. **Explanation of Design Alignment <> Meets:** A modular and

			distributed architecture enables the system to scale horizontally as needed.
Security			**Definition:** Security is the ability of the system to resist unauthorized attempts to access the system and/or its components. **Reason for Business Rank <> 3:** Security protects sensitive video metadata, transcripts, and user data from unauthorized access. **Explanation of Design Alignment <> Meets:** The system employs encryption, access controls, and secure API integrations to meet stringent security requirements.
Availability			**Definition:** Availability defines the days and hours the system is anticipated and/or required to be operational. **Reason for Business Rank <> 3:** Continuous system availability is crucial to support global users and uninterrupted service. **Explanation of Design Alignment <> Meets:** The deployment includes redundant systems and failover mechanisms to ensure 24/7 availability.
Supportability			**Definition:** Supportability defines the requirement that problems can be diagnosed and addressed quickly, with appropriately trained resources. **Reason for Business Rank <> 3:** Quick diagnosis and resolution of issues minimize downtime and maintain service quality. **Explanation of Design Alignment <> Meets:** Integrated logging, monitoring tools, and automated alerts are in place to support rapid troubleshooting.
Traceability			**Definition:** Traceability defines the ability to trace the history, application, or location of an item or activity by means of recorded identification. **Reason for Business Rank <> 3:** It is important for audit trails and debugging to have full traceability of data and changes. **Explanation of Design Alignment <> Meets:** Detailed logging of data transformations (e.g., transcript cleaning, comment

			filtering) ensures complete traceability throughout the processing pipeline.
User Usability			**Definition:** User Usability defines how easy the interface is to use to successfully complete tasks, including clear navigation, terminology, and error messaging. **Reason for Business Rank <> 3:** A user-friendly interface directly impacts user satisfaction and adoption. **Explanation of Design Alignment <> Meets:** The design features an intuitive dashboard with clear visualizations for video summaries, sentiment analysis, and structured feedback.

2. Quality Attributes- Development Qualities

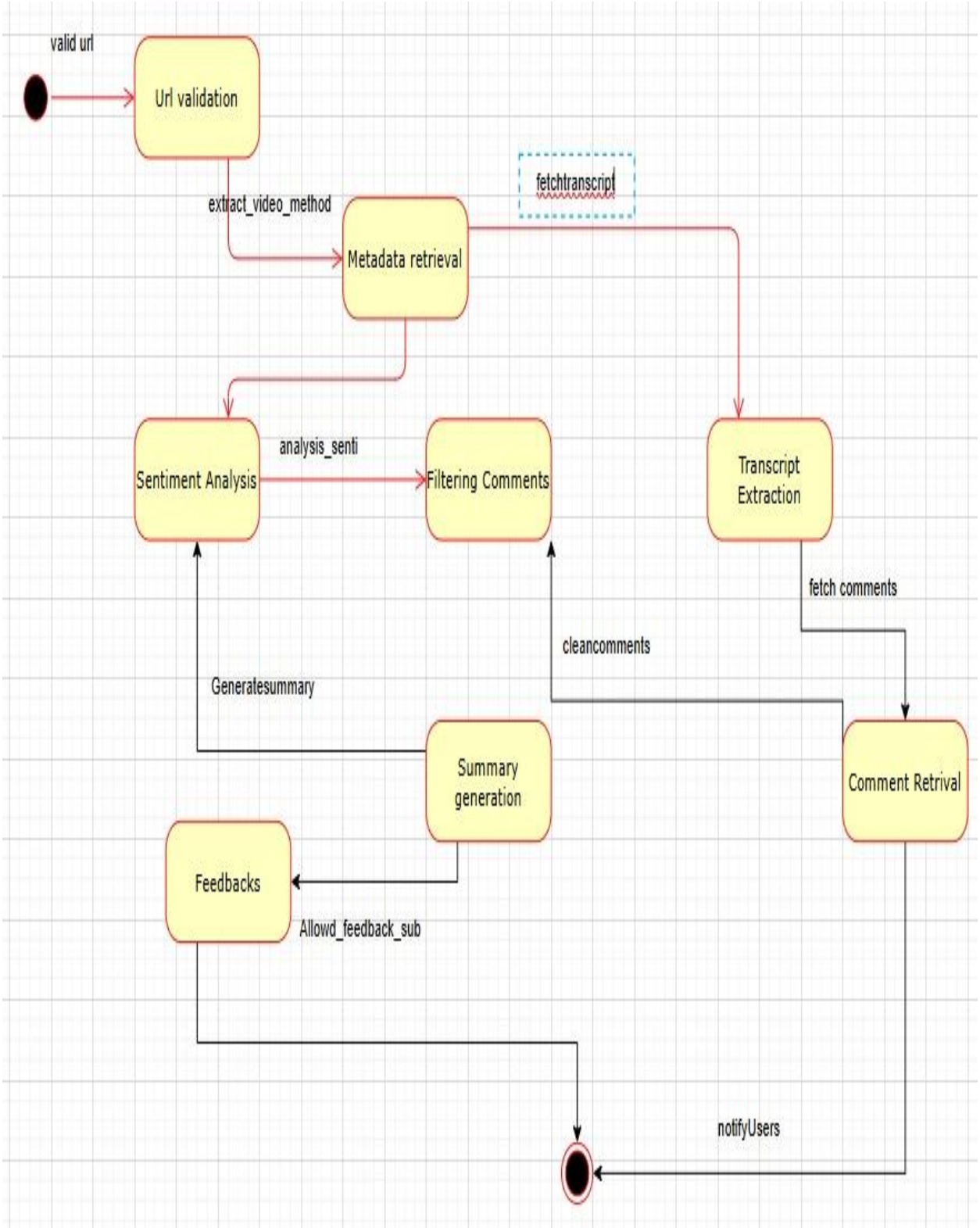
Attributes	**Bus Rank**	**Design Align**	**Run Time Quality Attributes Documents**
Reusability			**Definition:** The reusability of the system is the ability to reuse portions of the system in other applications. **Reason for Business Rank <> 3:** High reusability reduces development time and cost by allowing components to be leveraged in future projects. **Explanation of Design Alignment <> Meets:** The system is designed with modular components (data retrieval, transcript cleaning, sentiment analysis) that can be reused across different applications.
Extensibility			**Definition:** Extensibility defines the degree to which the system can be extended to incorporate new functionality without changing the fundamental architecture, design, or core source code. **Reason for Business Rank <> 3:** Extensibility is essential to accommodate evolving business needs and integration with additional data sources. **Explanation of Design Alignment <> Meets:** Published interfaces such as APIs and metadata definitions allow for seamless extension of system capabilities.

Modifiability			**Definition:** Modifiability (maintainability) defines the efficiency of making enhancements or maintenance changes. **Reason for Business Rank <> 3:** High modifiability minimizes disruption and cost when updating the system. **Explanation of Design Alignment <> Meets:** A well-structured, documented, and modular codebase allows for quick modifications without affecting core functionality.
Ease of Integration			**Definition:** Ease of Integration defines how easily the application or its components can be integrated or packaged with new applications. **Reason for Business Rank <> 3:** Seamless integration is key to interoperability with external systems and tools. **Explanation of Design Alignment <> Meets:** The system uses standardized, RESTful APIs and common data formats to facilitate integration.
Phase-In "Ability"			**Definition:** Phase In "ability" (subset-ability) defines how the application accommodates implementation/rollout schedules without impacting overall design. **Reason for Business Rank <> 3:** Phased deployment minimizes risk and allows for early user feedback. **Explanation of Design Alignment <> Meets:** The architecture supports incremental deployment, enabling features to be released in stages.
Conceptual Integrity			Definition: Conceptual Integrity refers to the ability of the architecture to communicate a clear, concise vision in line with the enterprise architecture. Reason for Business Rank <> 3: Ensures a unified approach and consistency across the system, which is vital for long-term maintainability. Explanation of Design Alignment <> Meets: Consistent design principles and a unified processing pipeline ensure all components work cohesively.

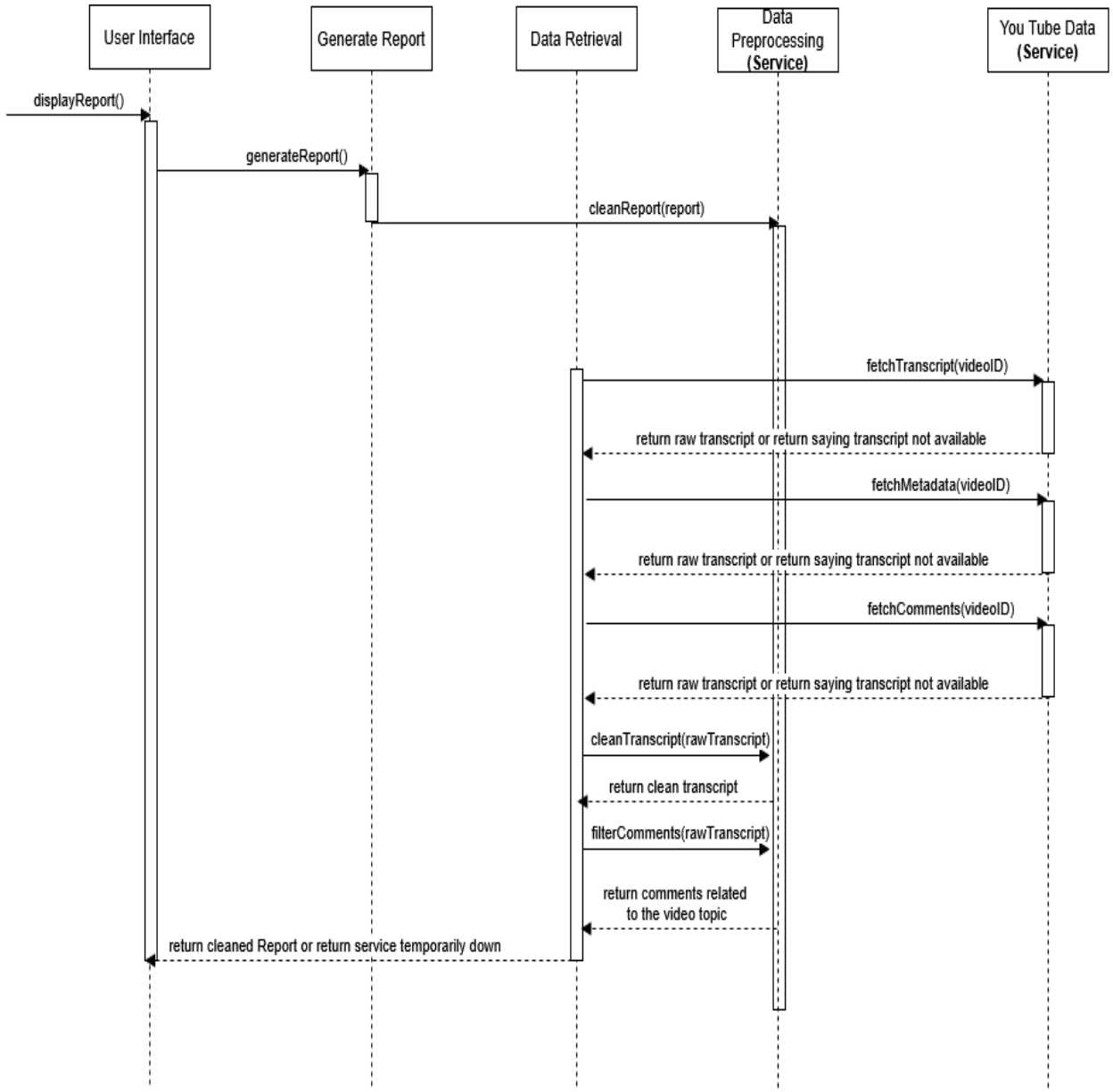
Testability			**Definition:** Testability describes the ability to test each component independently (unit tests) as well as in an integrated environment. **Reason for Business Rank <> 3:** High testability reduces risk of defects and ensures reliable system performance with each update. **Explanation of Design Alignment <> Meets:** A comprehensive suite of automated tests is integrated into the CI/CD pipeline to verify all critical components.
Portability			**Definition:** Portability measures the ease with which the system can be moved to different platforms (hardware, operating systems, virtualization, etc.). **Reason for Business Rank <> 3:** Portability provides flexibility in deployment and minimizes vendor lock-in. **Explanation of Design Alignment <> Meets:** The system is developed using cross-platform frameworks and containerization, ensuring compatibility across environments.
Developer Complexity			**Definition:** Developer Complexity is how easy it is for developers to understand and create the system. **Reason for Business Rank <> 3:** Lower complexity accelerates onboarding and reduces development errors. **Explanation of Design Alignment <> Meets:** The architecture is simplified with clear module boundaries and detailed documentation, reducing the overall complexity.
Buildability			**Definition:** Buildability defines the degree to which the system can be compiled, packaged, archived, and deployed using automated tools and standard processes. **Reason for Business Rank <> 3:** Efficient build processes accelerate time-to-market and ensure consistency. **Explanation of Design Alignment <> Meets:** The integration of an automated CI/CD pipeline and build automation tools streamlines deployment and minimizes errors.

SPRINT 3

State diagram



SOA SERVICES



ID	Product Backlog	Estimation	Priority
1	As a User, I can enter a YouTube video URL on the home page and click “Search” to initiate video data retrieval (metadata, transcript, and comments).	3	1
18	Extract actionable “What’s Working” feedback from filtered comments using the ChatGPT API.	1	18
19	Extract actionable “Needs Improvement” feedback from filtered comments using the ChatGPT API.	1	19
20	Generate a structured feedback report categorizing “What’s Working” and “Needs Improvement” and store the report.	2	20
21	As a User, I can view a results page that displays the compiled video metadata (e.g., title, description, duration).	3	21
22	As a User, I can view a results page that displays the generated video summary.	2	22
23	As a User, I can view a results page that displays detailed sentiment analysis results of comments.	2	23
24	As a User, I can view a results page that displays the structured feedback report categorizing “What’s Working” and “Needs Improvement”.	3	24

Implementation of Sprint 3:

- Integrated Tkinter UI with HomePage (ID 1) for input and ResultsPage to display analysis (ID 21, 22, 23, 24)
- Applied prompt engineering to improve structure and clarity of AI-generated outputs (ID 18, 19)
- Auto-displayed video summary, sentiment %, and bullet-point feedback in scrollable UI
- Implemented background threading with loading popup for smooth UX
- Saved feedback as timestamped .txt report in /reports folder (ID 20)

Retrospective: What went well

- Prompt engineering improved accuracy and structure of AI outputs
- Gemini-based bullet-point feedback displayed clearly with bolded subtopics
- UI handled empty inputs, bad URLs, and API errors without crashing
- Background thread + animated loading popup kept UI responsive
- Feedback report saved correctly in local folder

Retrospective: Challenges

- Regex spam filtering still misses nuanced promotional comments (data_preprocessor.py)
- API keys remain hardcoded in main.py (Sprint 2 issue not resolved)
- Sentiment classification slows down at ~3000 comments (batching not implemented)
- Bold text workaround required due to tk.Label formatting limitations

Output:

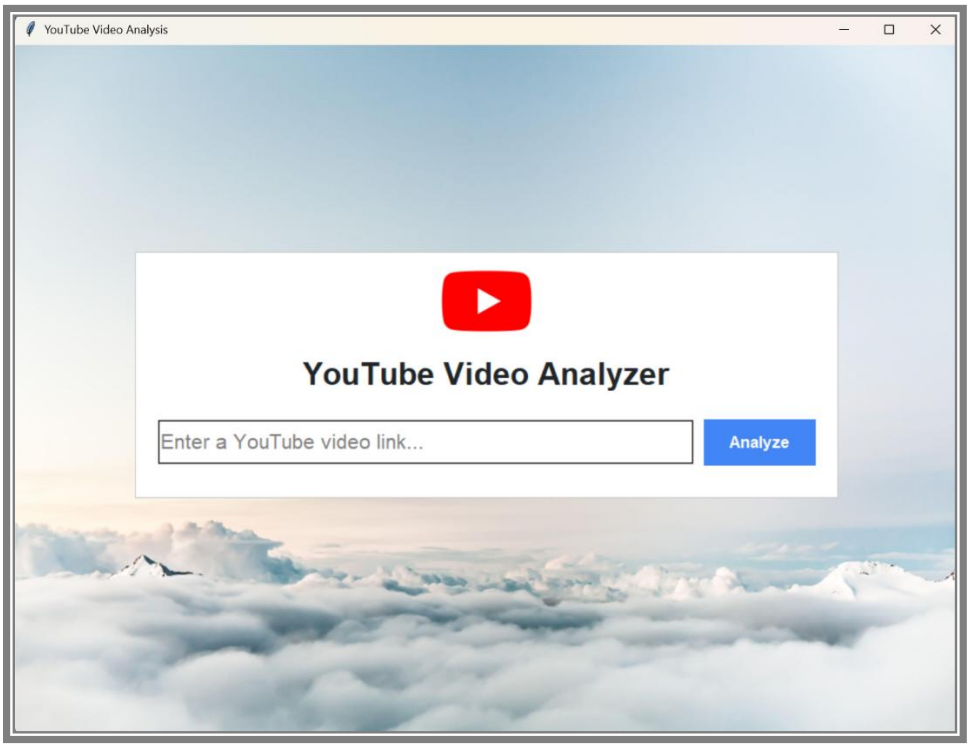


Fig: Normal Demo of project

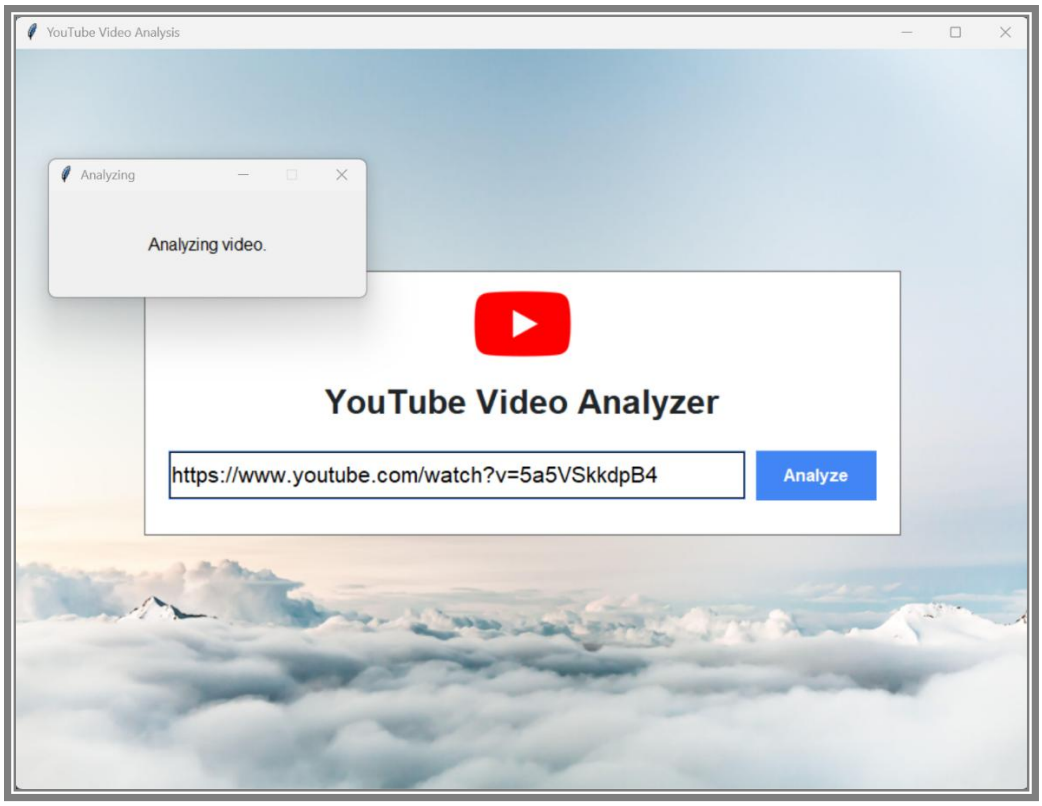


Fig: Home page when analyzing

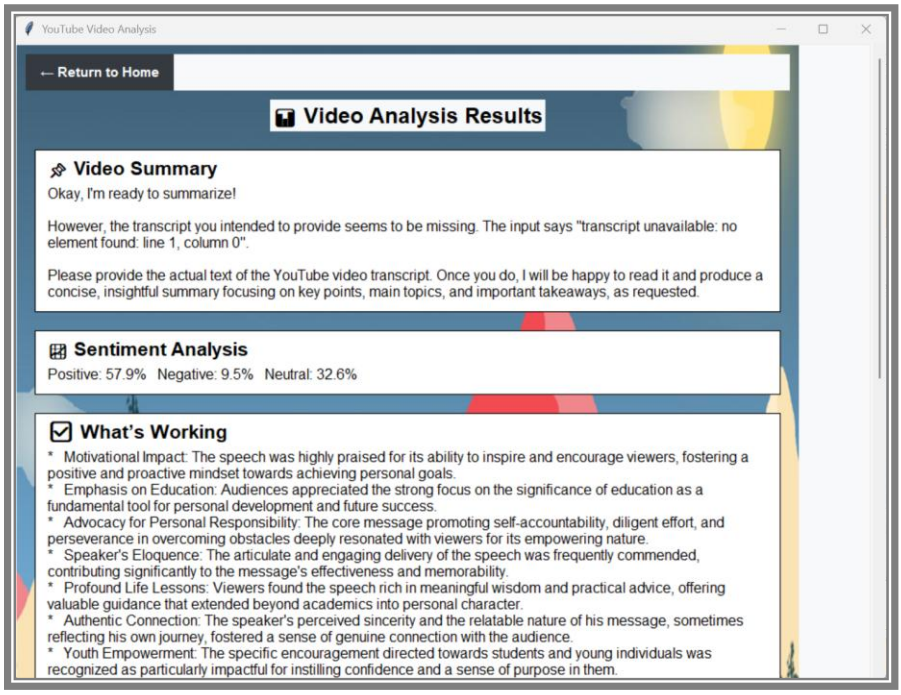


Fig: Results page part one

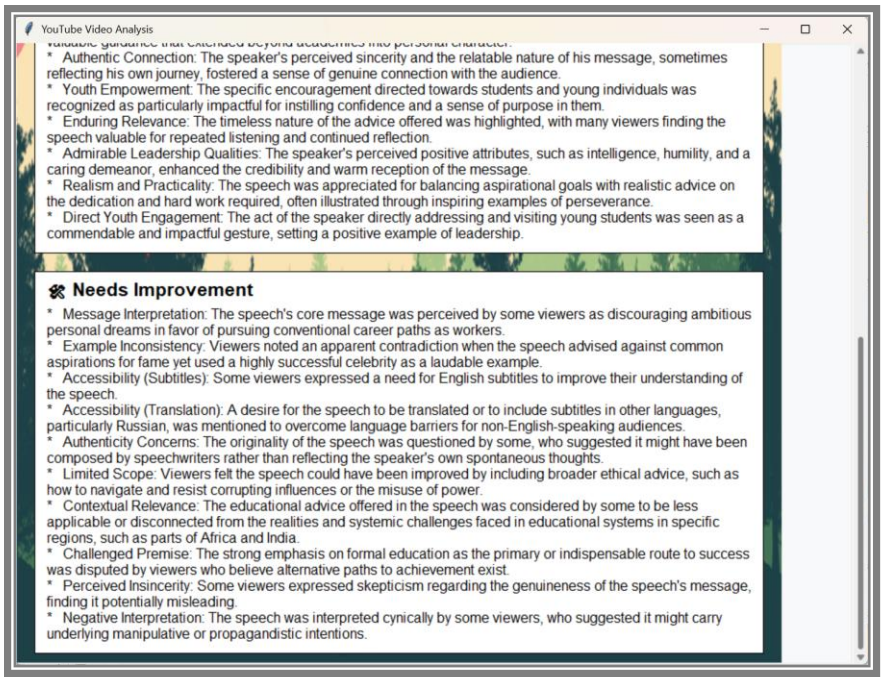
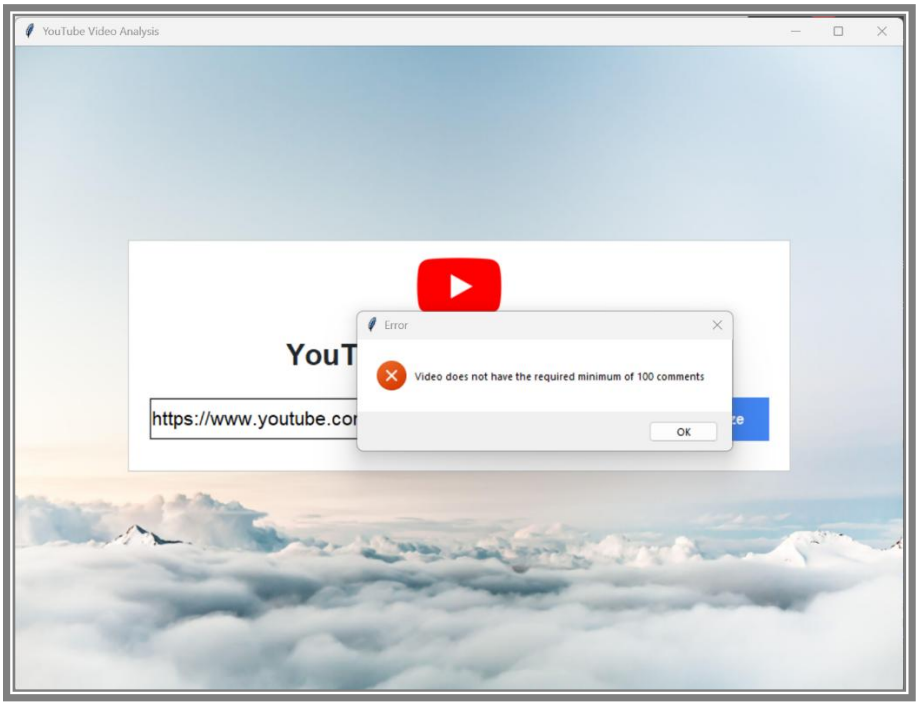


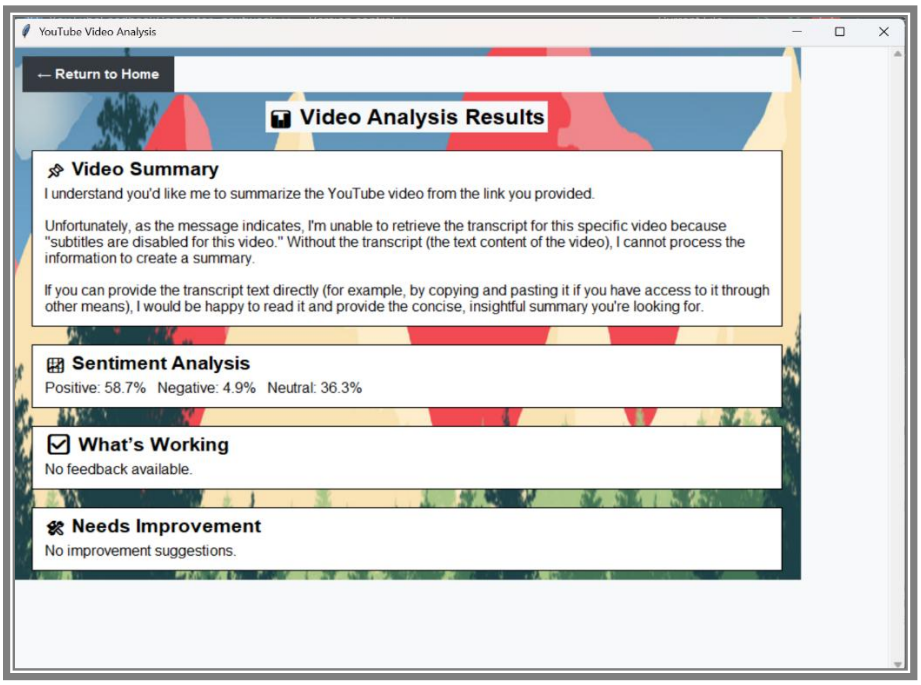
Fig: Results page part two

Test Cases

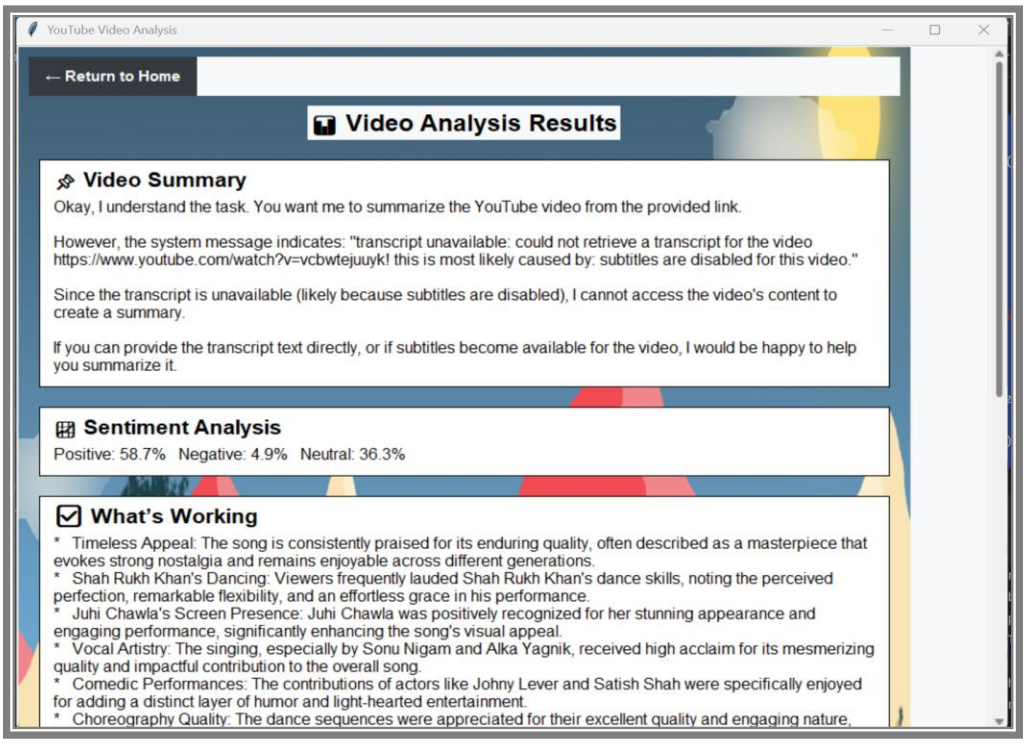
Test case 1: If comments < 100



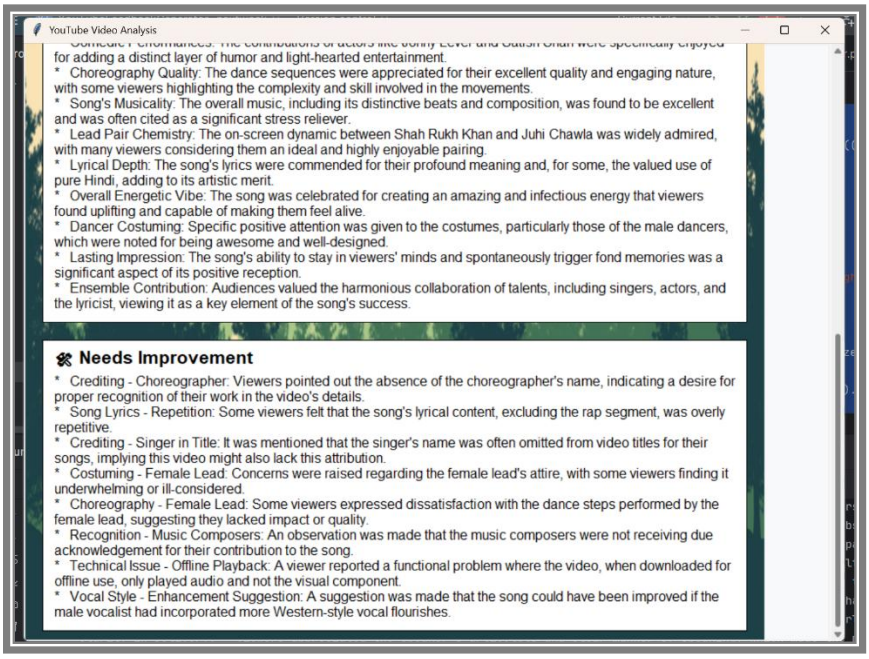
Test case 2: When there is no transcript



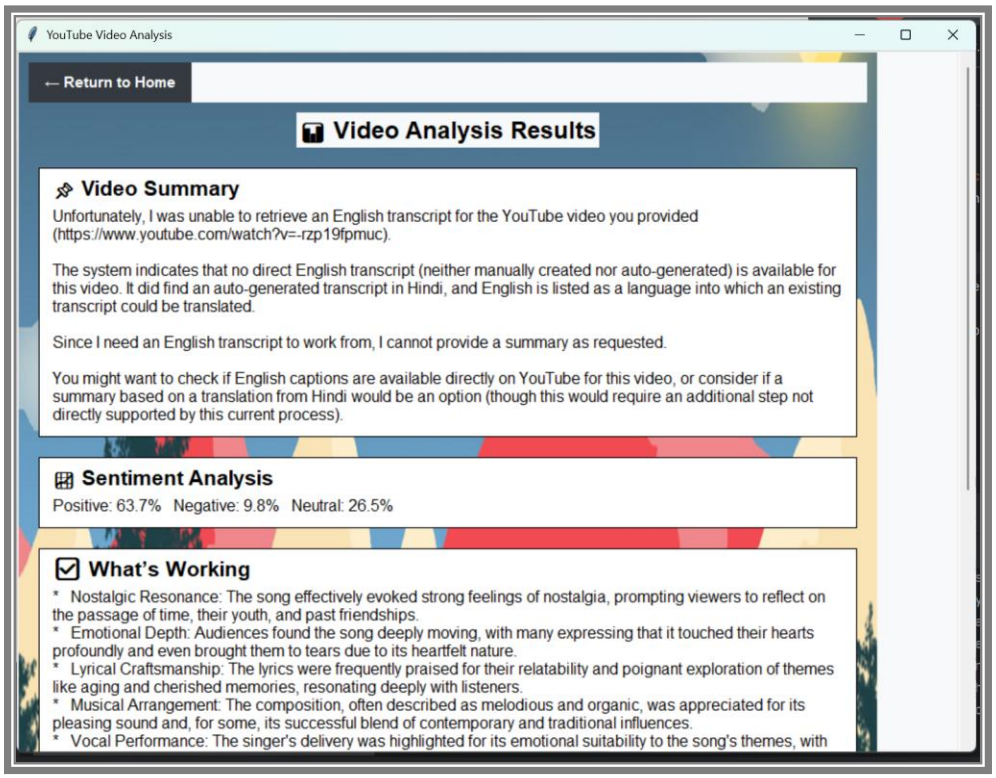
Test case 3: When no transcript but comments part one



Test case 4: When no transcript but comments part two



Test case 5: when it has foreign transcript



Program Code:

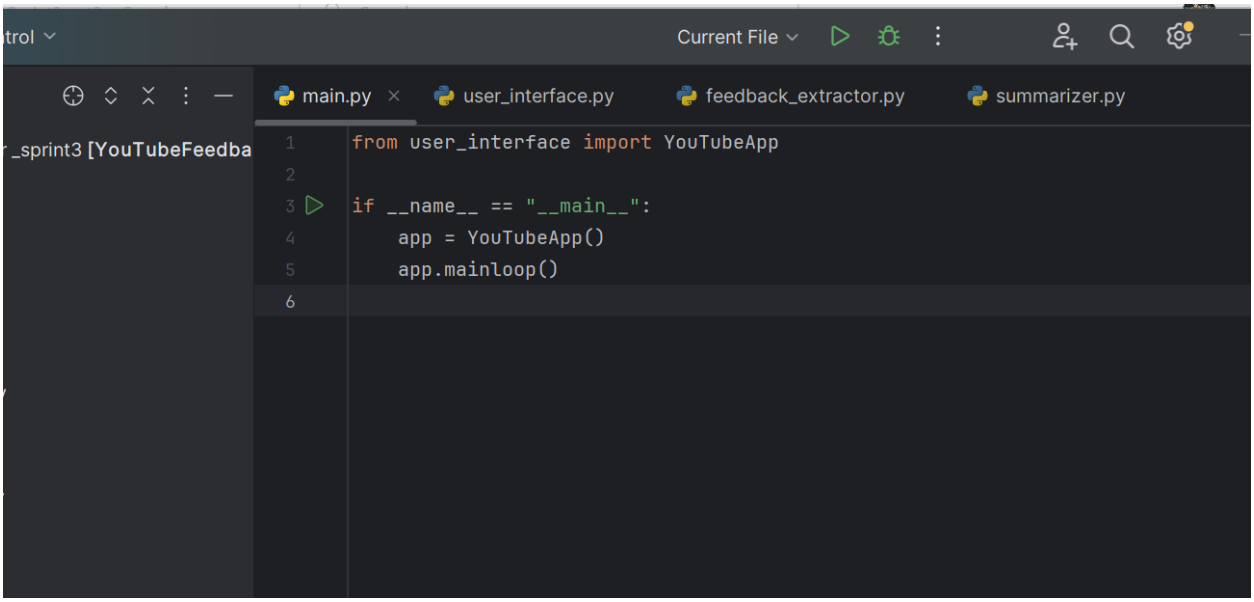


Fig: main.py

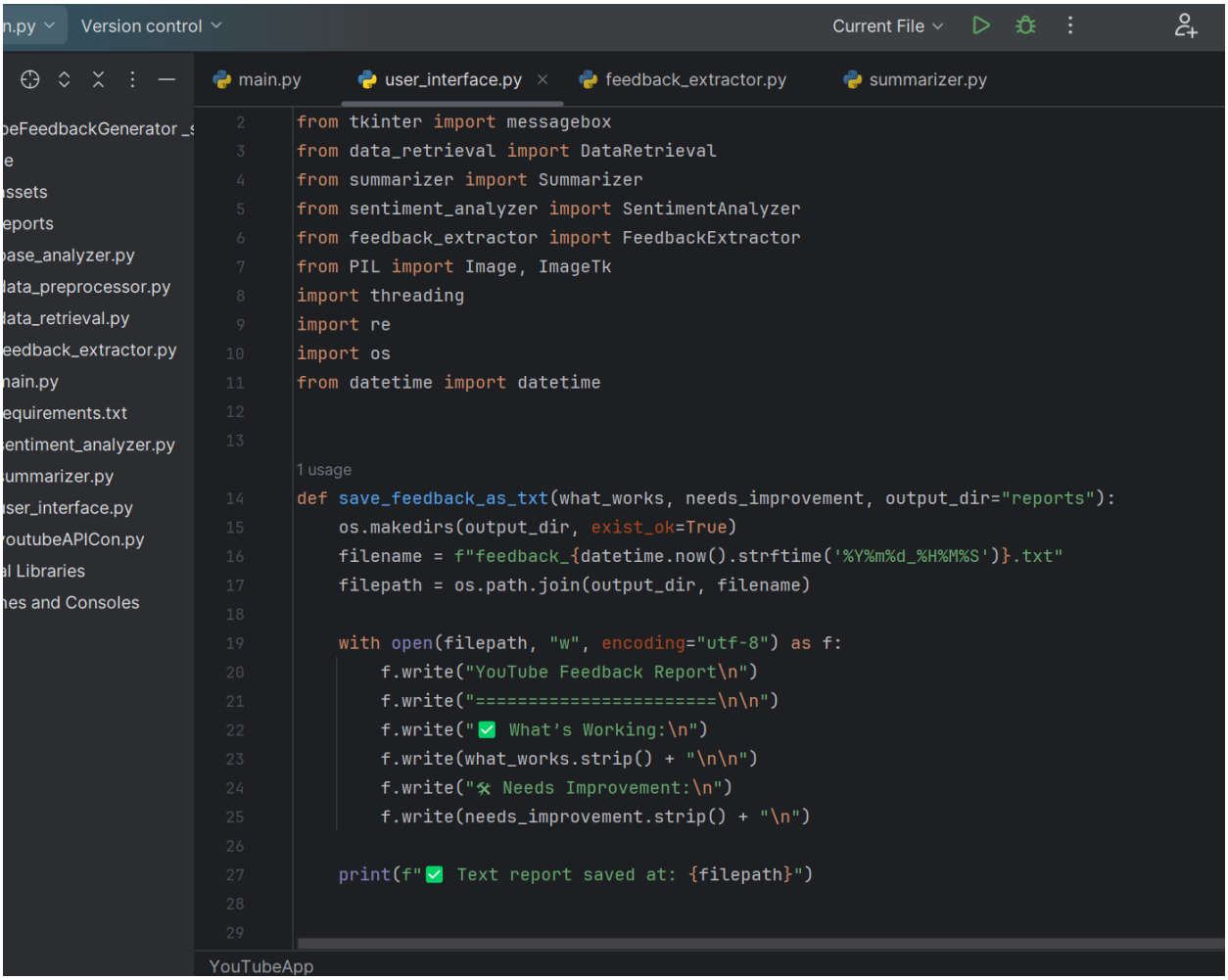


Fig: user_interface.py part 1

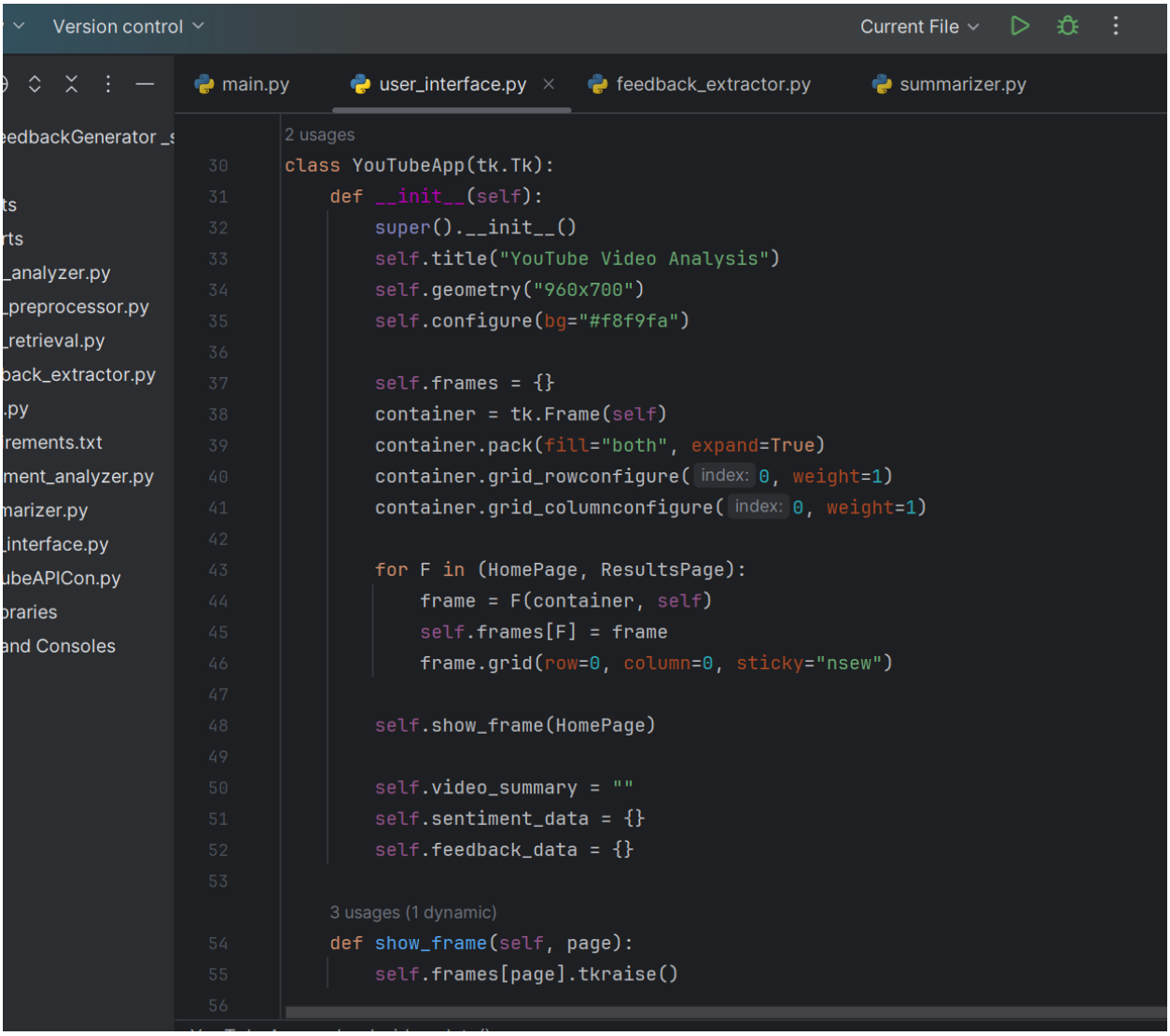


Fig: user_interface.py part 2

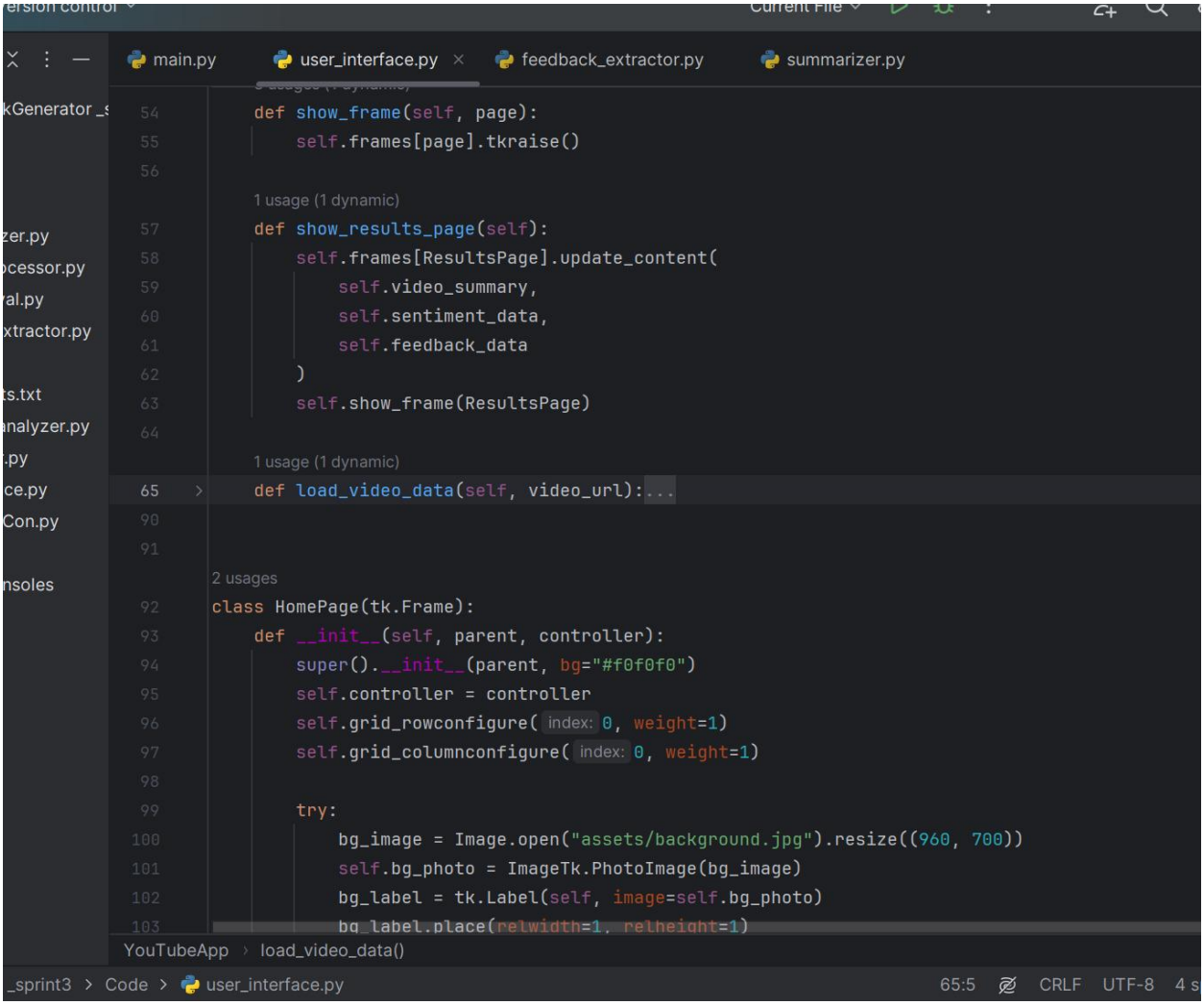


Fig: user_interface.py part 3

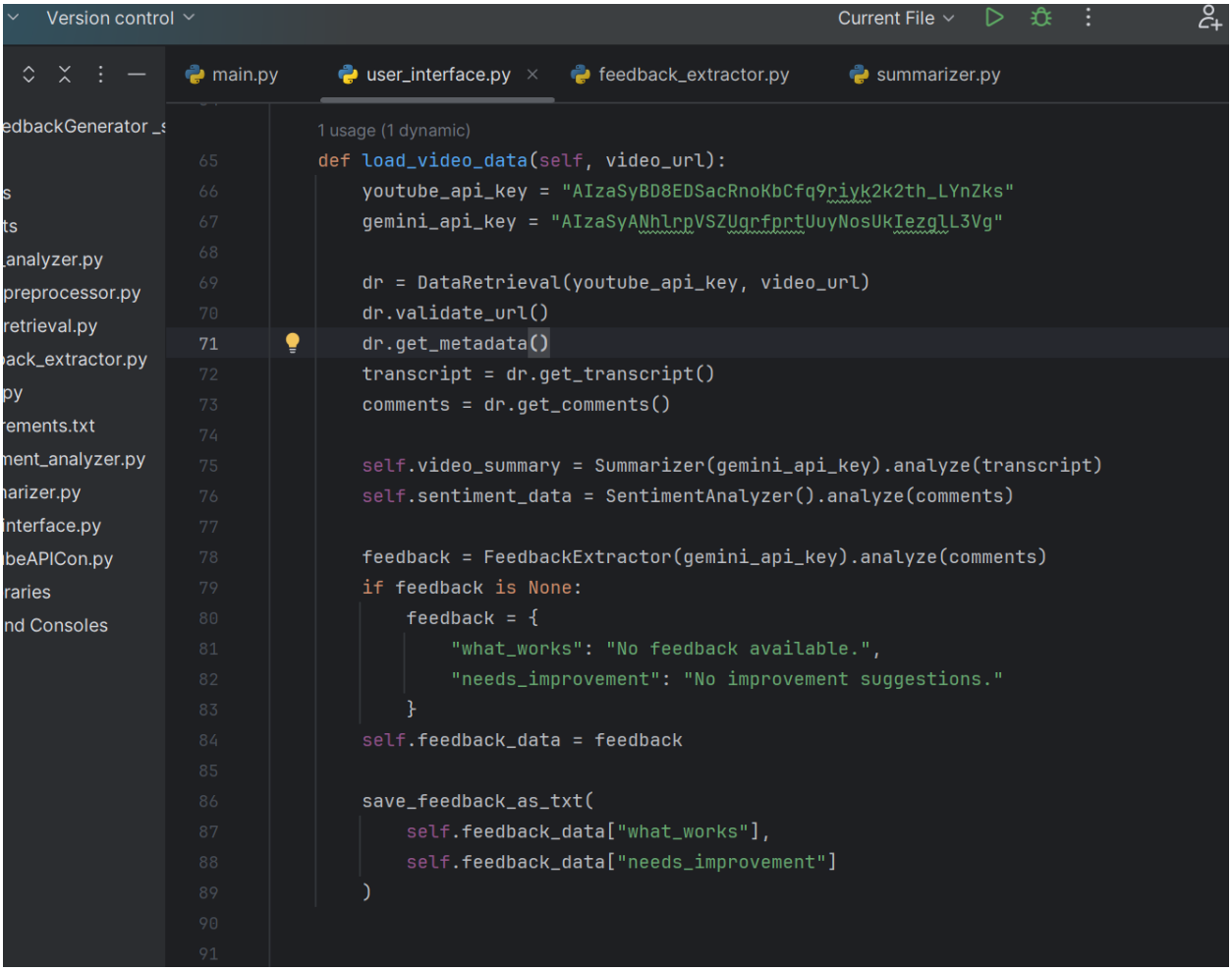


Fig: user_interface.py load_video_data function

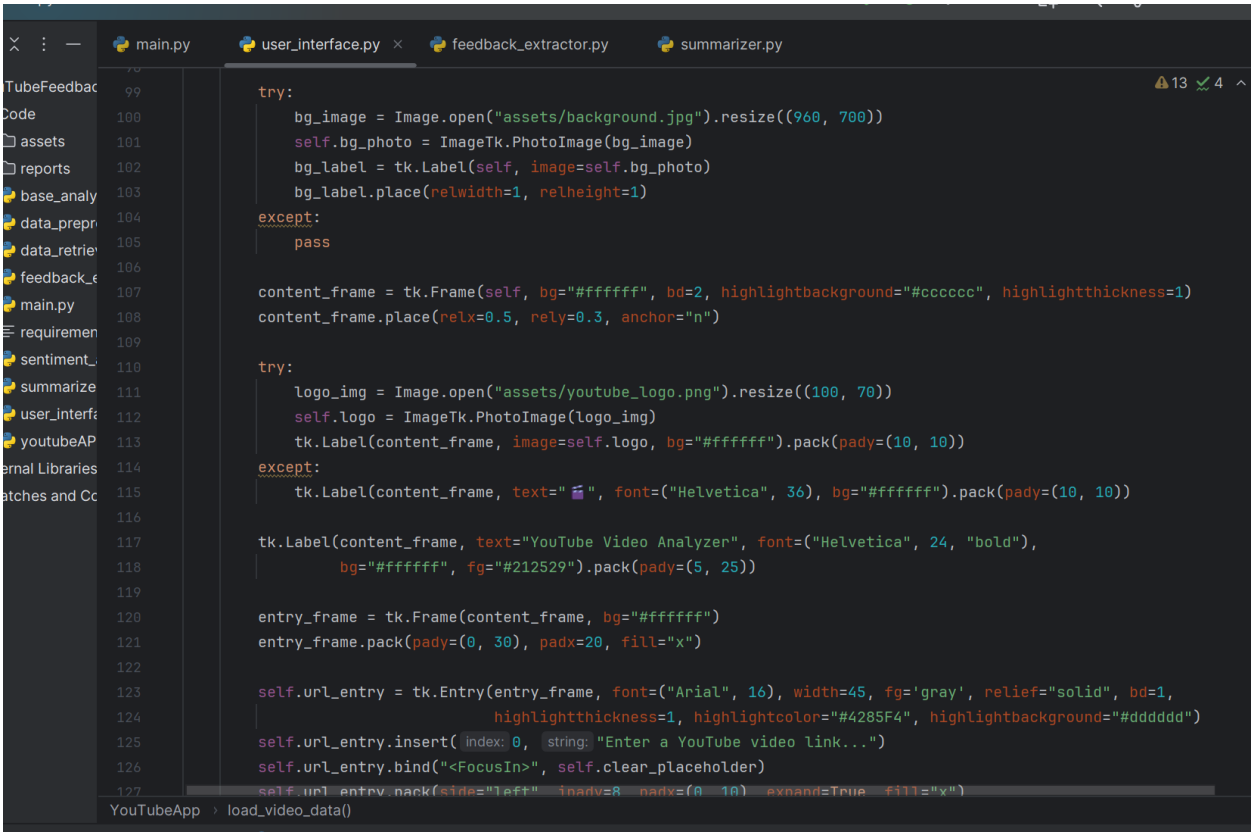


Fig: user_interface.py part 4

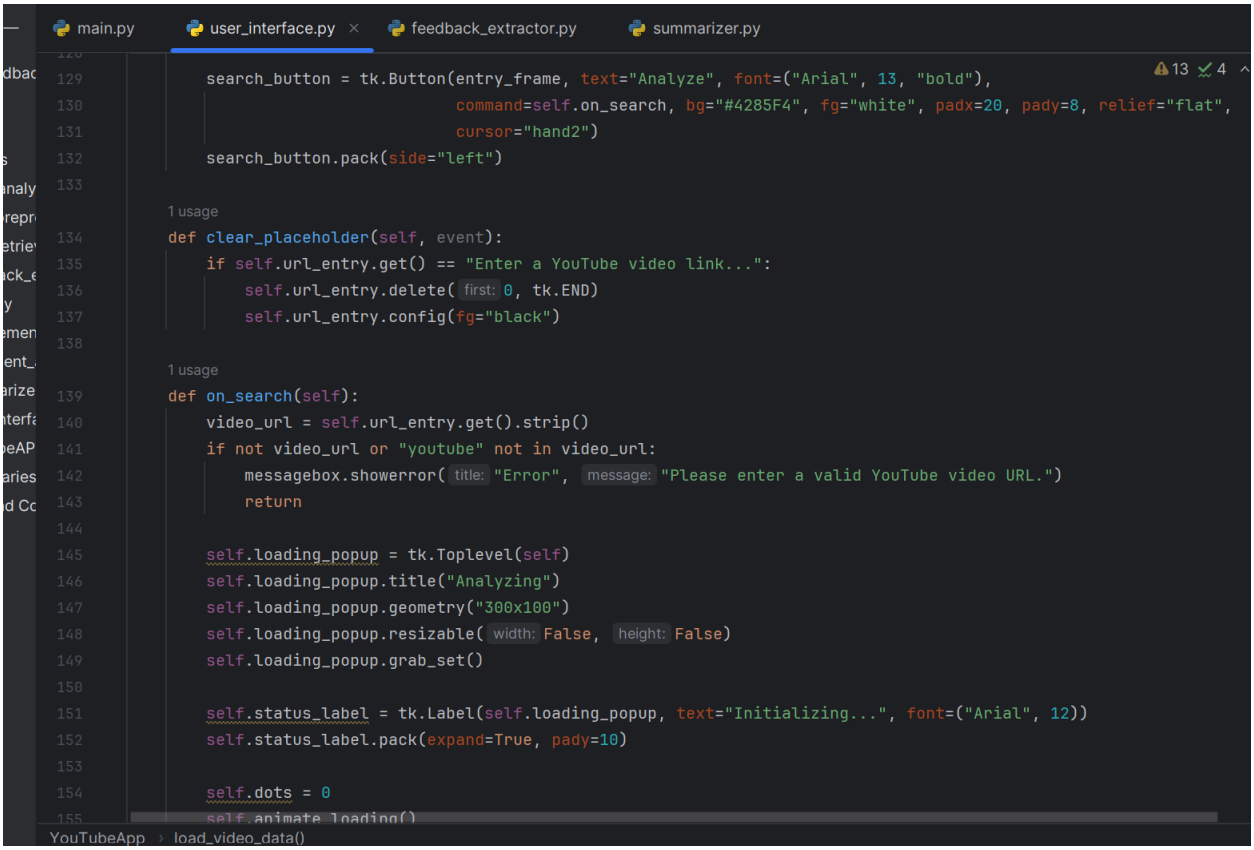


Fig: user_interface.py part 5

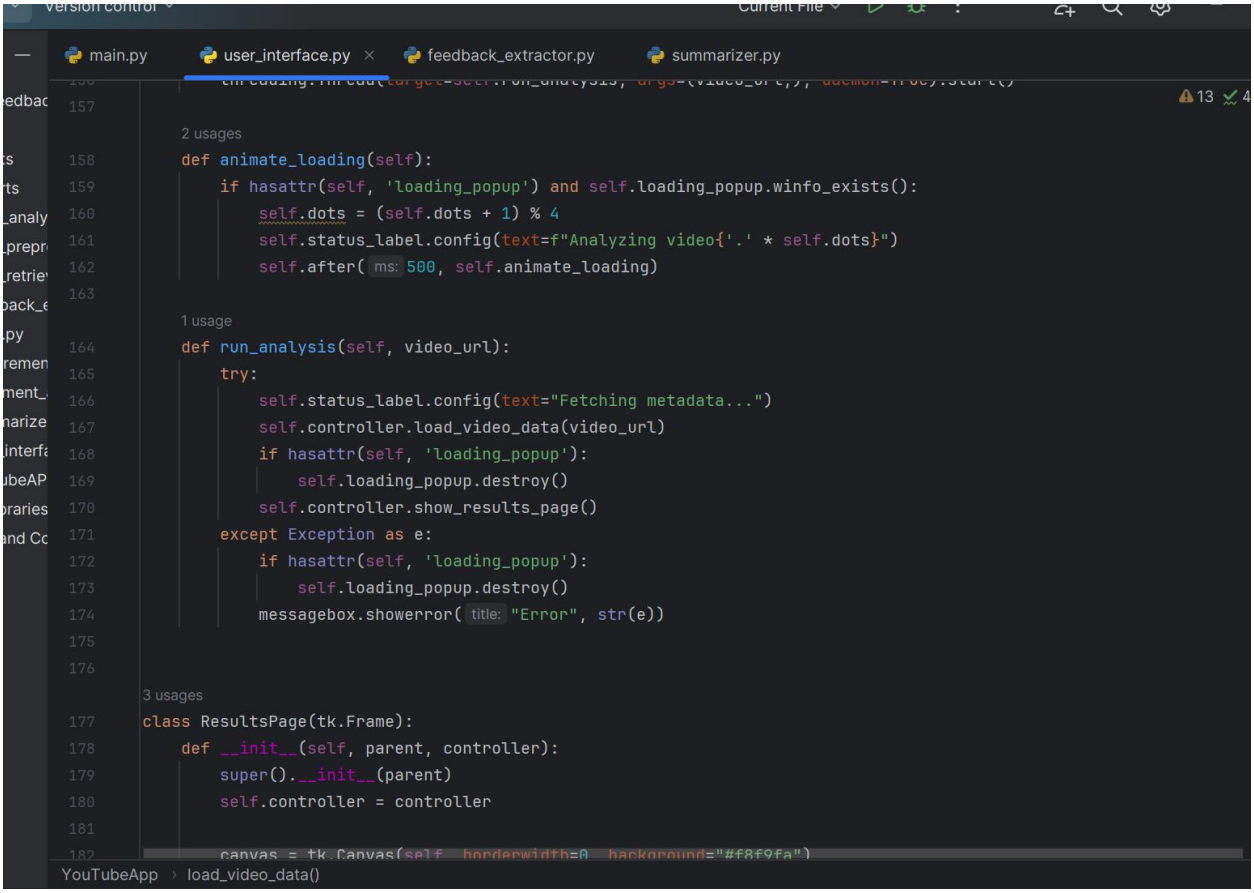


Fig: user_interface.py part 6

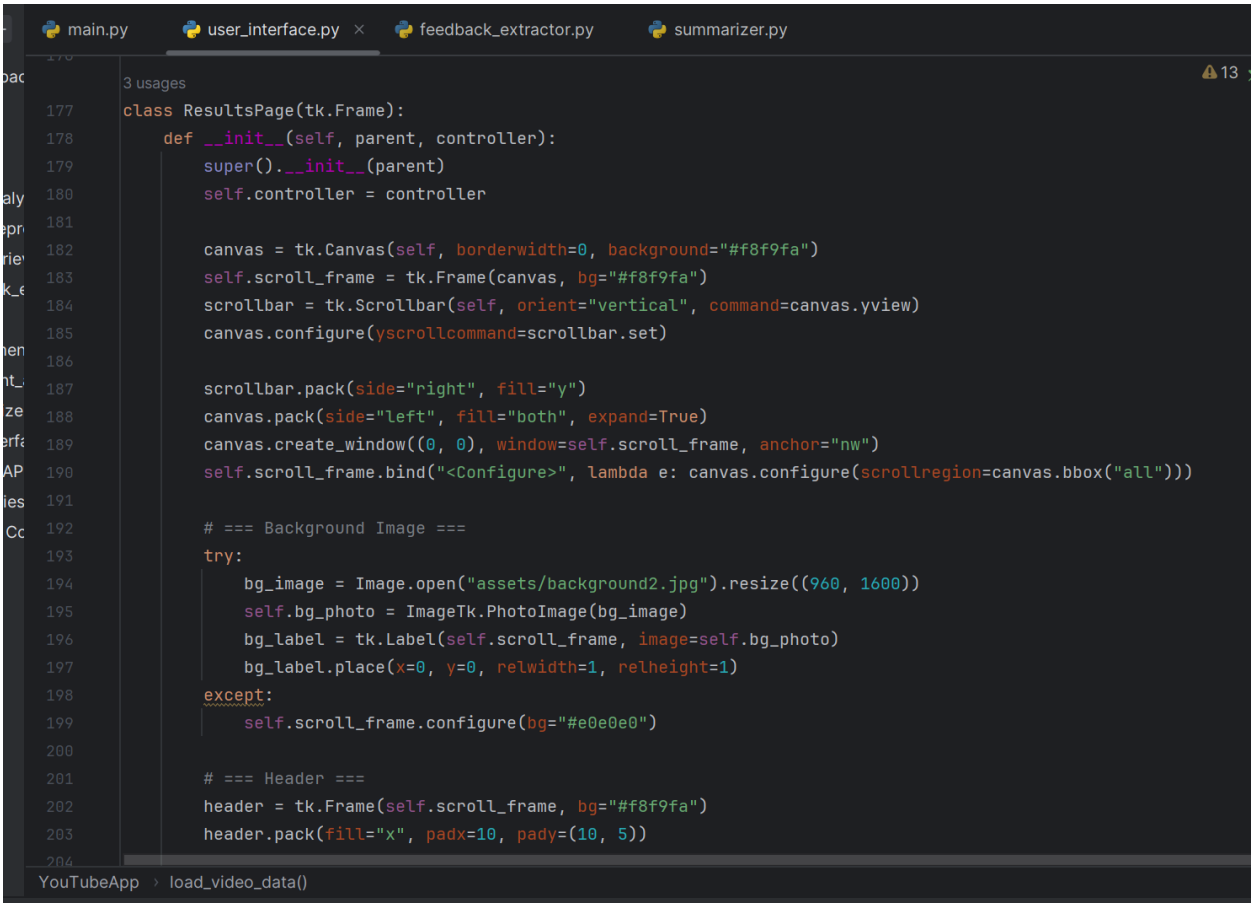


Fig: user_interface.py part 7

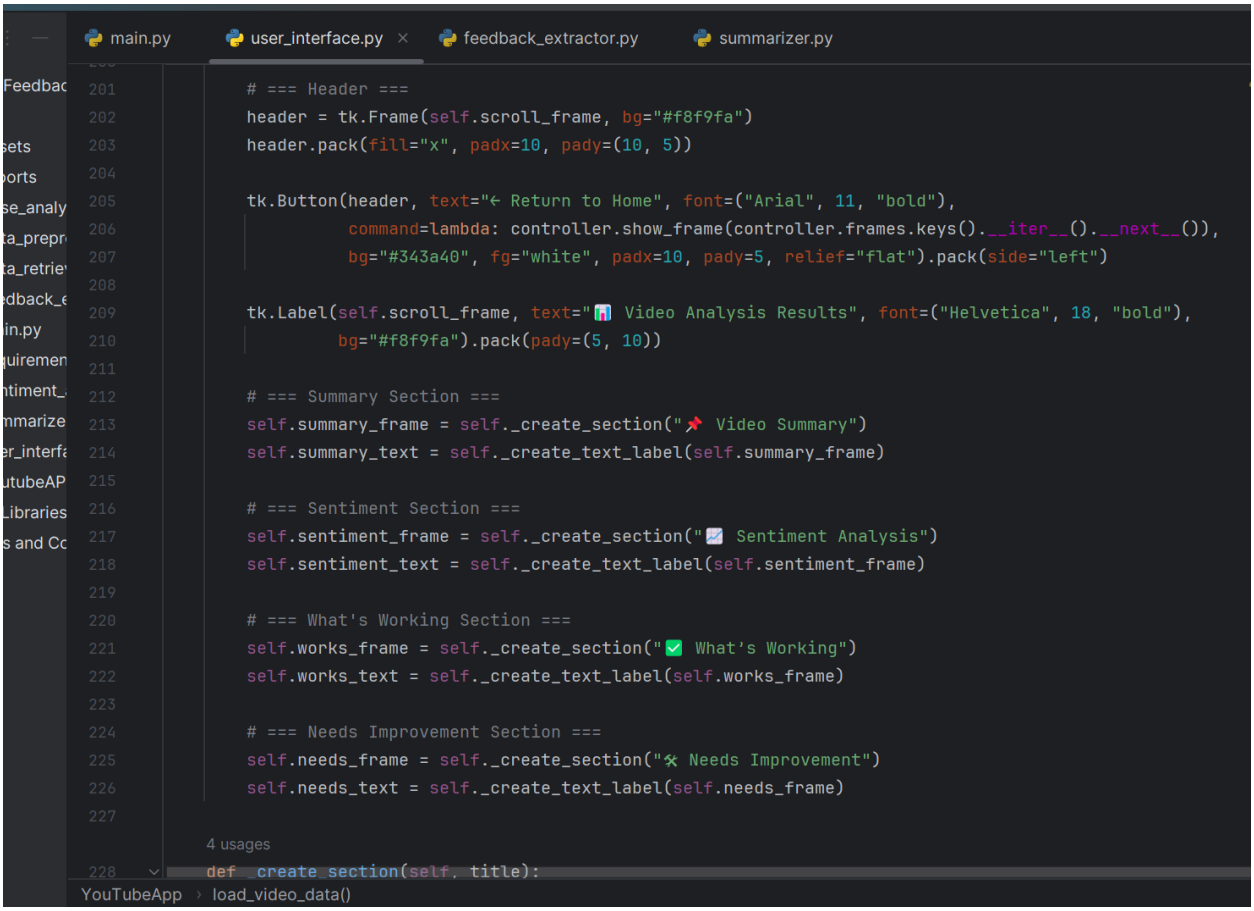


Fig: user_interface.py part 8

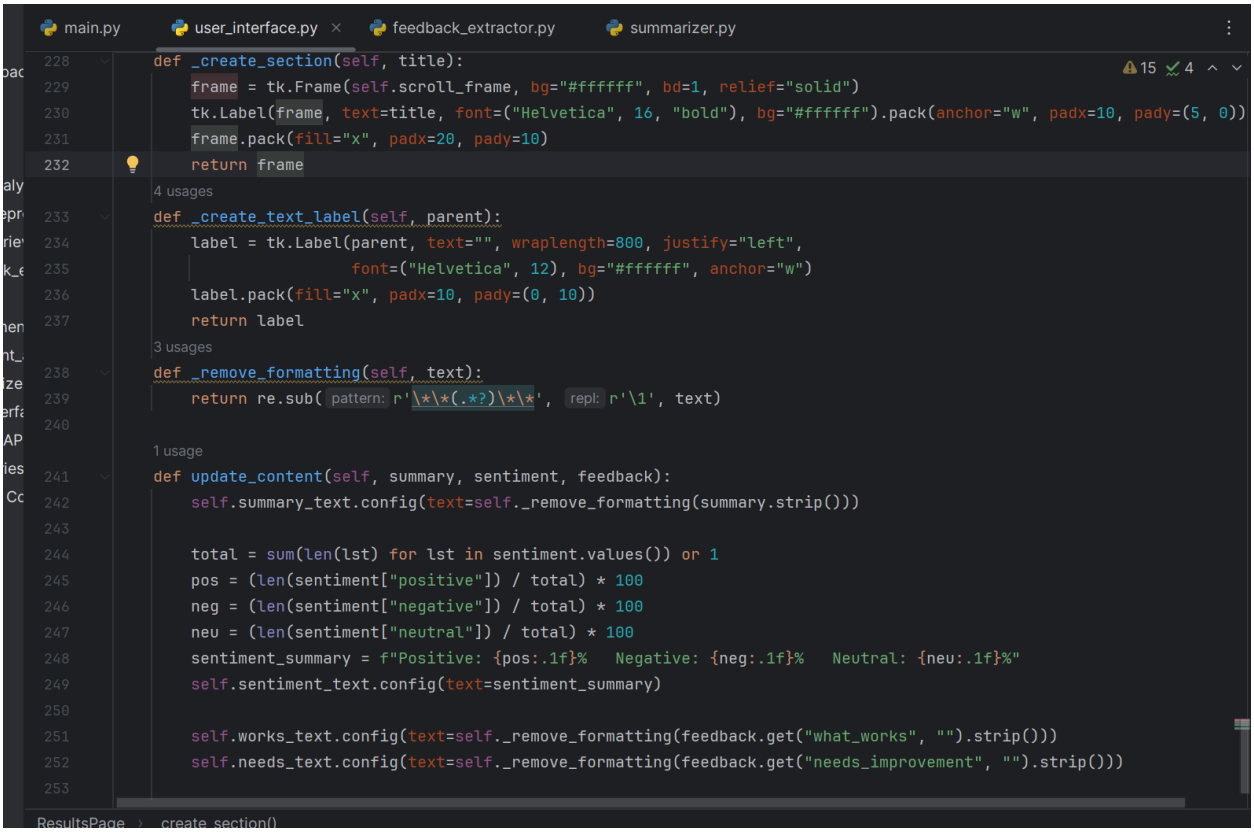


Fig: user_interface.py part 9

main.py | user_interface.py | feedback_extractor.py | summarizer.py

```python
import google.generativeai as genai
from typing import List, Dict, Optional
from base_analyzer import BaseAnalyzer


4 usages
class FeedbackExtractor(BaseAnalyzer):
    """Processes comments using Gemini AI to extract 'What's Working' and 'Needs Improvement' feedback."""

    def __init__(self, api_key: str):
        super().__init__(api_key)
        genai.configure(api_key=self.api_key)

    WORKS_PROMPT = (
        "You are a professional video content analyst. Given the following YouTube comments about a video, "
        "analyze and summarize"
        "the most praised elements with insight. Your task is to extract deeper feedback trends that reflect what the "
        "audience appreciated."
        "List these as bullet points. Each bullet point must start with a bolded subtopic (e.g., **Clarity**) "
        "followed by a concise 1-2 sentence"
        "explanation of why this was positively received. Do not quote comments directly. Do not provide introductory "
        "or concluding sentences."
        "Do not group points or use themes. Be objective and informative. Note only use insights from comments, "
        "not your simulation.\n\nComments:\n{comments}"
    )

    IMPROVEMENT_PROMPT = (
        "You are a professional video content analyst. Given the following YouTube comments about a video, "
        "analyze and summarize"
```

Fig: Feedback code part 1

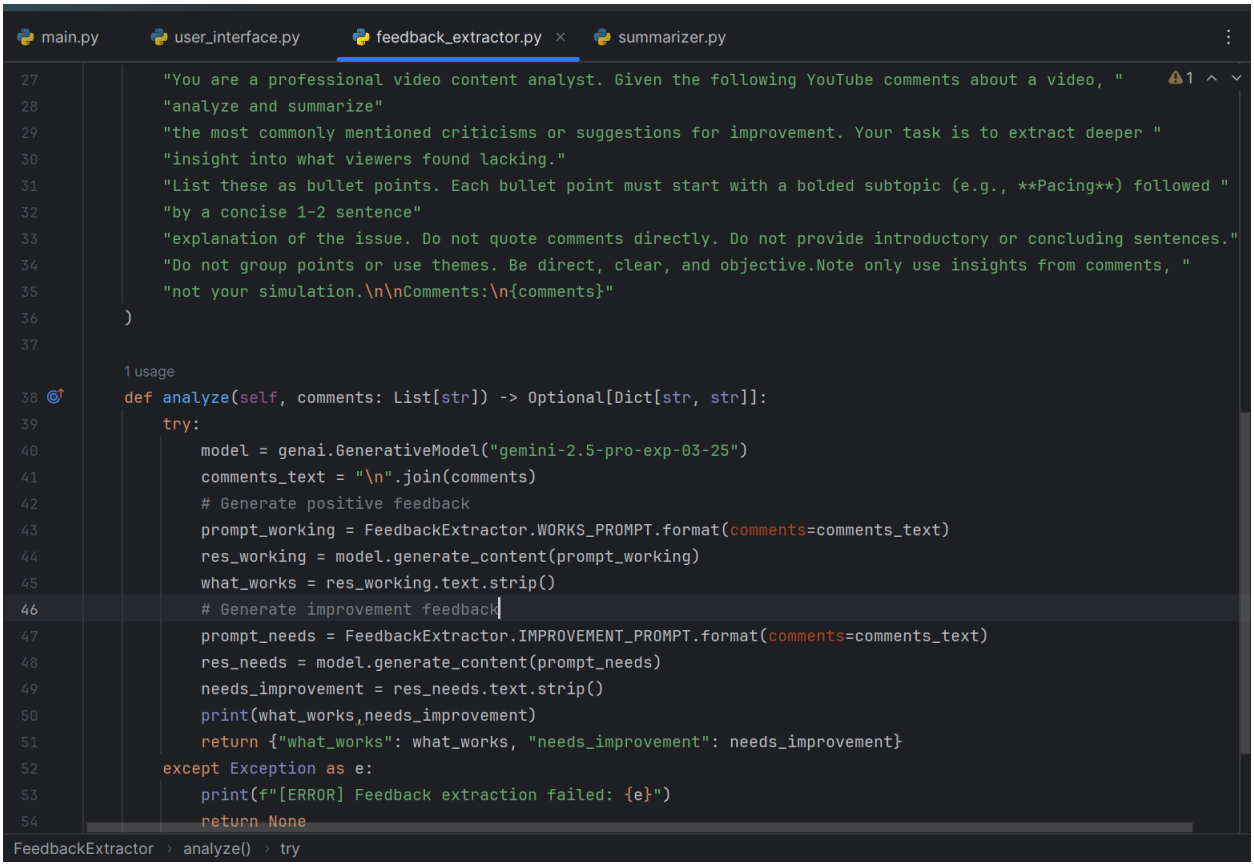


Fig: Feedback code part 2

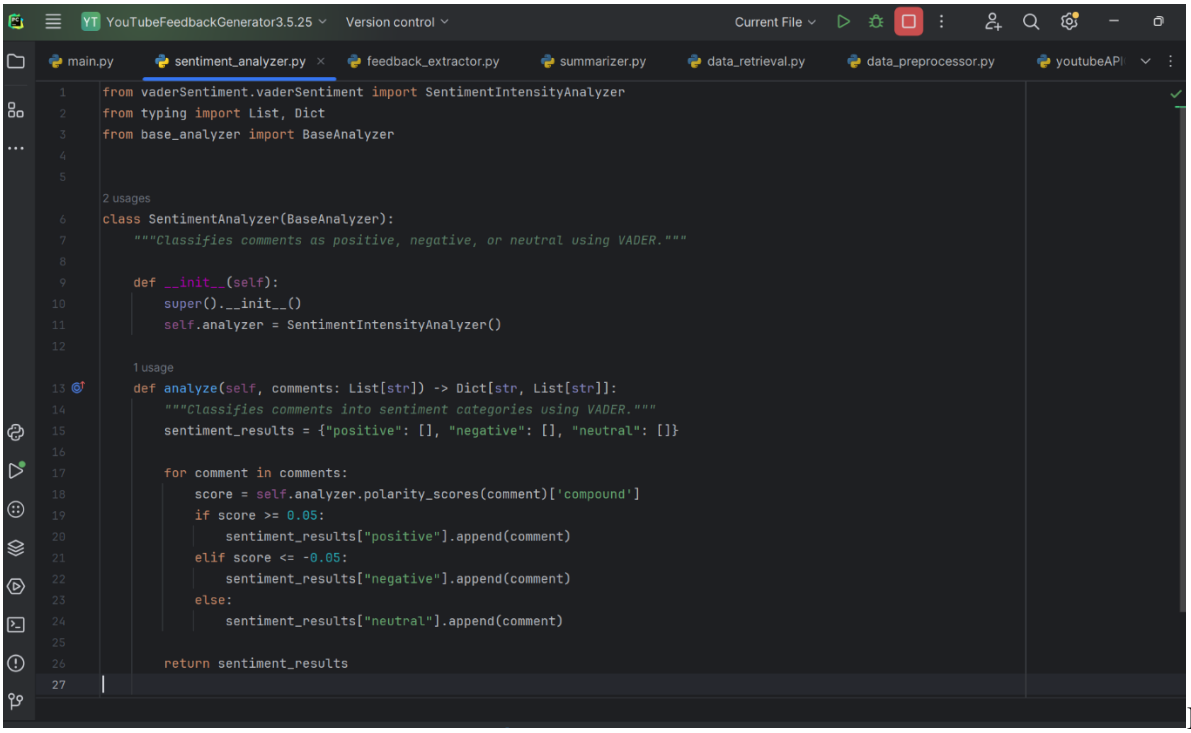


Fig:

Fig: sentiment analysis code

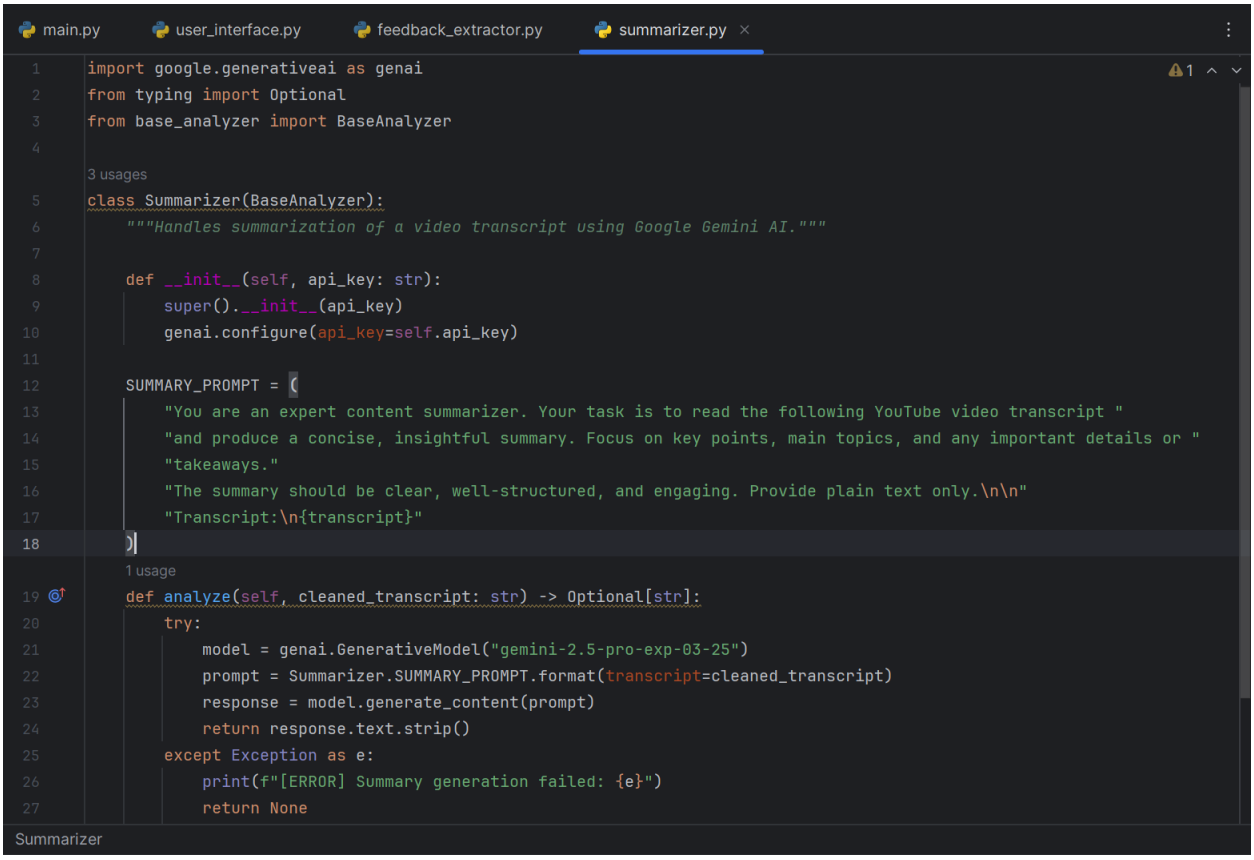


Fig: summarizer code

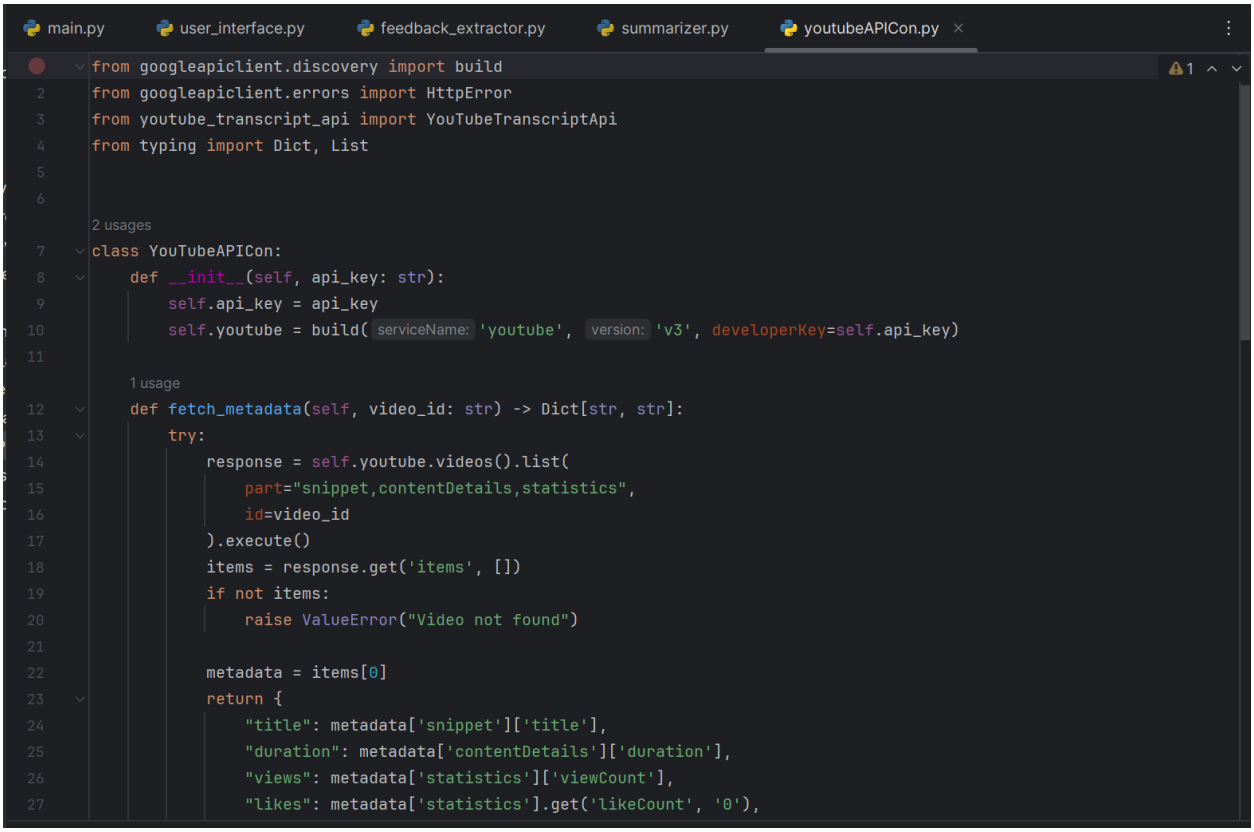


Fig: YouTubeAPICon class

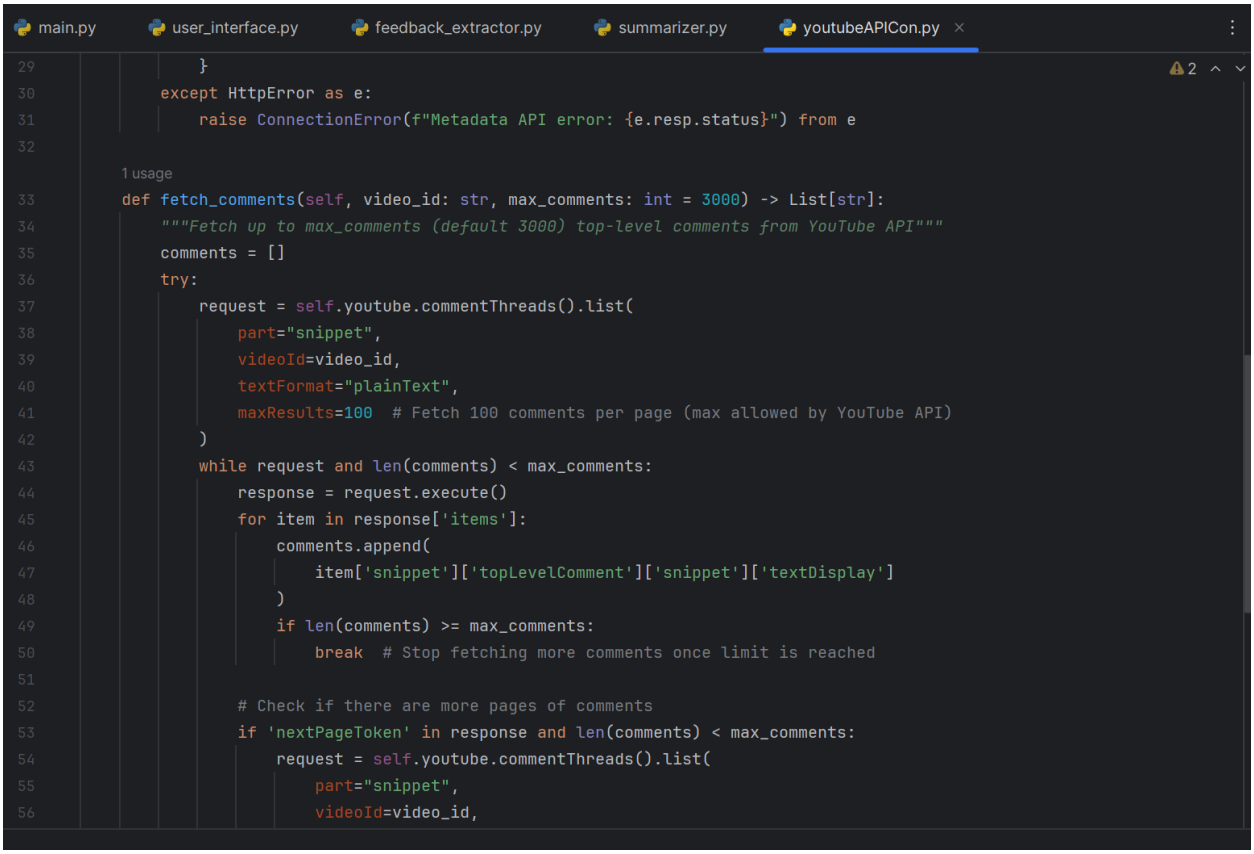


Fig: YouTubeAPICon class part 2

main.py | user_interface.py | feedback_extractor.py | summarizer.py | youtubeAPICon.py

```
            # Check if there are more pages of comments
            if 'nextPageToken' in response and len(comments) < max_comments:
                request = self.youtube.commentThreads().list(
                    part="snippet",
                    videoId=video_id,
                    textFormat="plainText",
                    maxResults=100,
                    pageToken=response['nextPageToken']
                )
            else:
                break  # No more pages, exit loop

        return comments
    except HttpError as e:
        raise ConnectionError(f"Comments API error: {e.resp.status}") from e

1 usage
def fetch_transcript(self, video_id: str) -> str:
    try:
        transcript_list = YouTubeTranscriptApi.get_transcript(video_id)
        transcript = ' '.join([entry['text'] for entry in transcript_list])
        return transcript
    except Exception as e:
        return f"Transcript unavailable: {str(e)}"
```

Fig: YouTubeAPICon class part 3

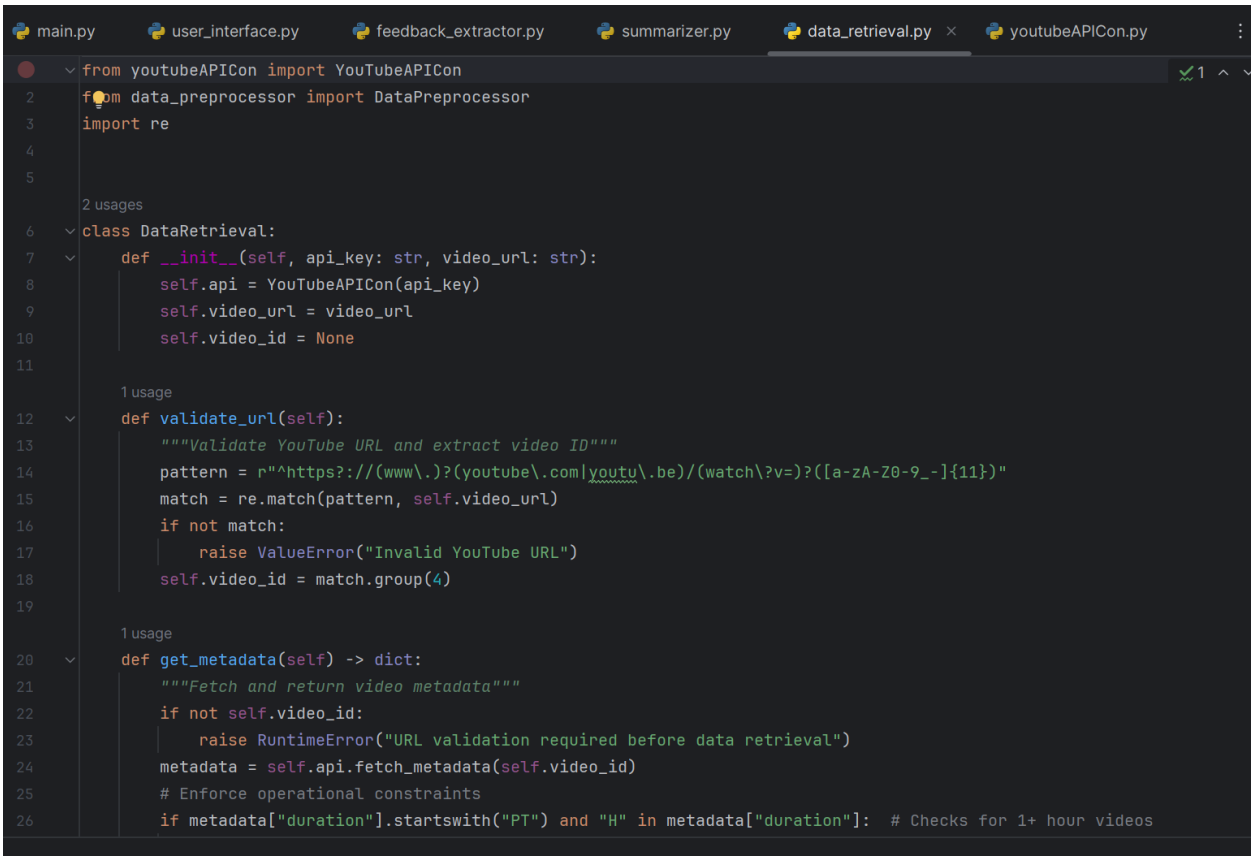


Fig: Data Retrieval .py part 1

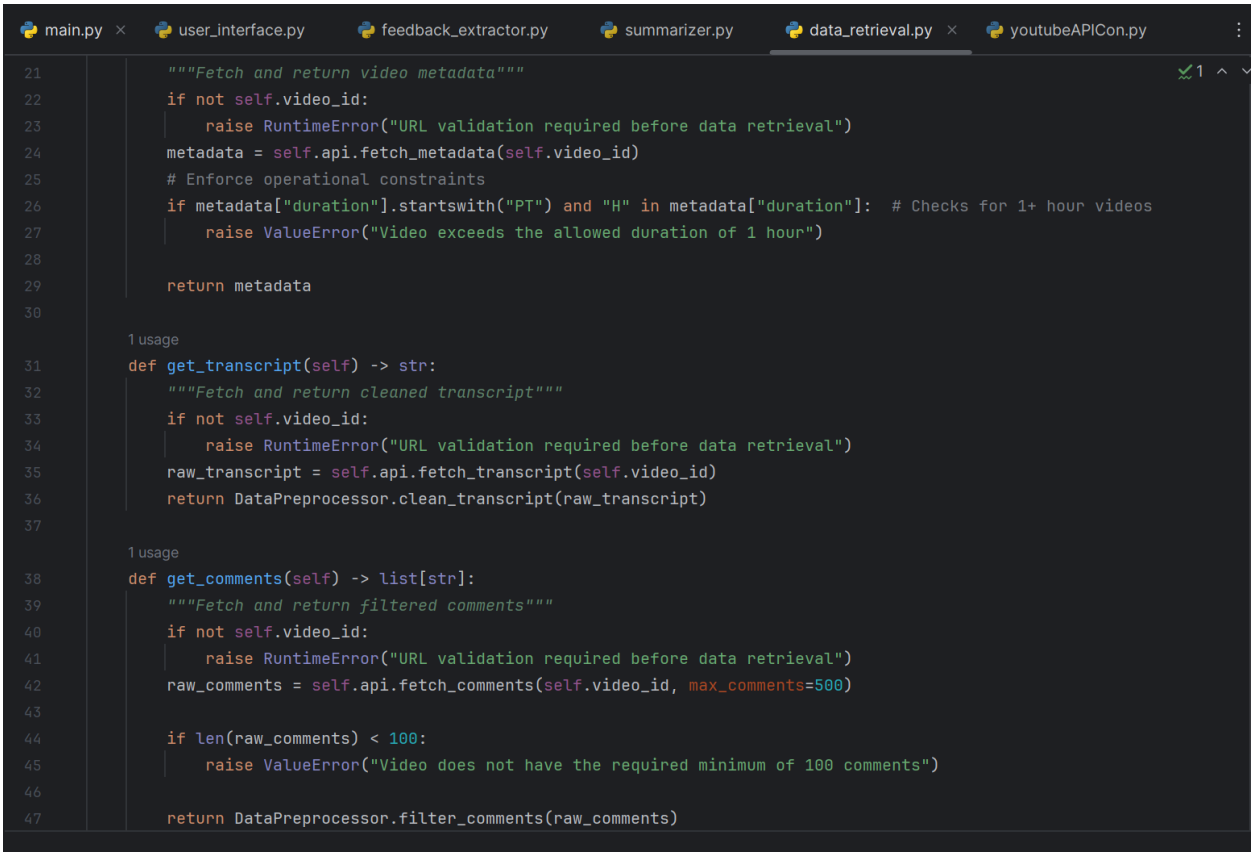


Fig: data retrieval.py part 2

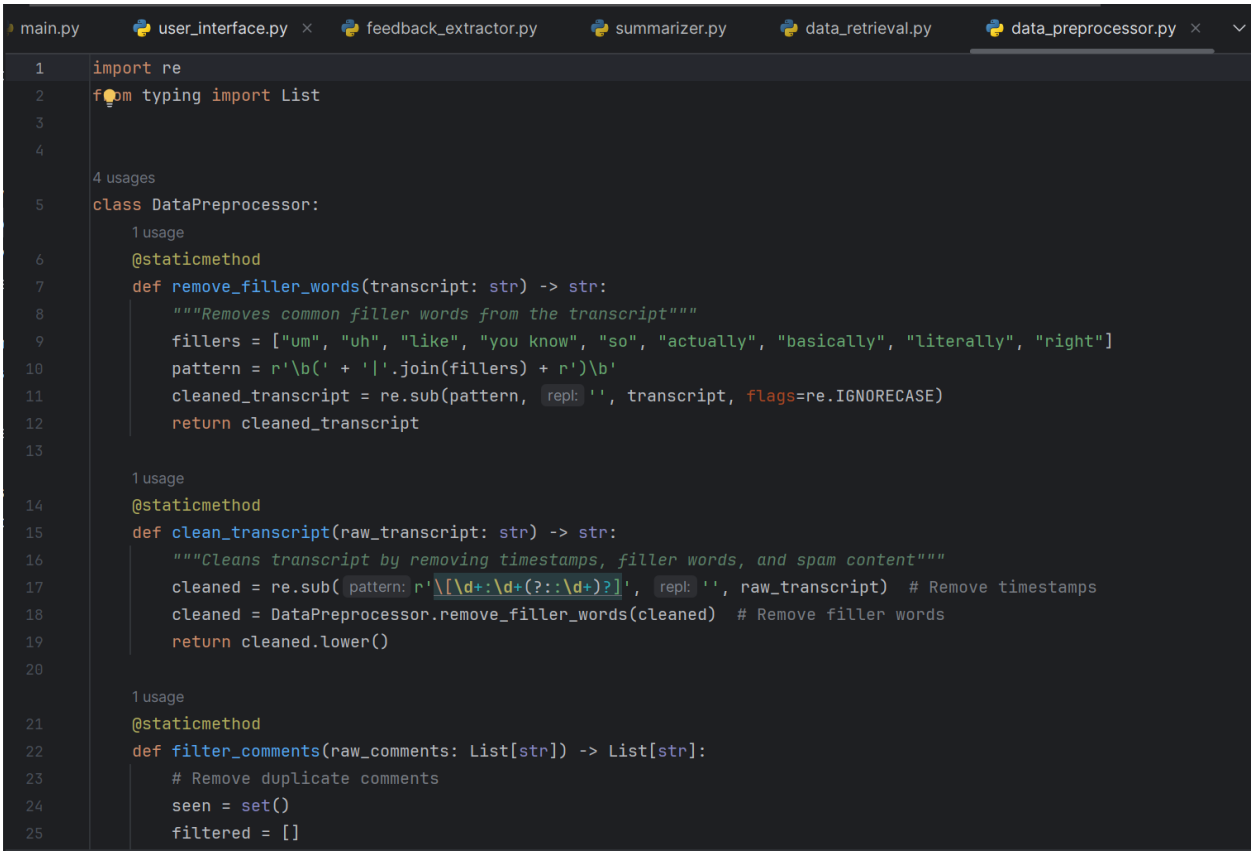


Fig: data_preprocessor.py part 1

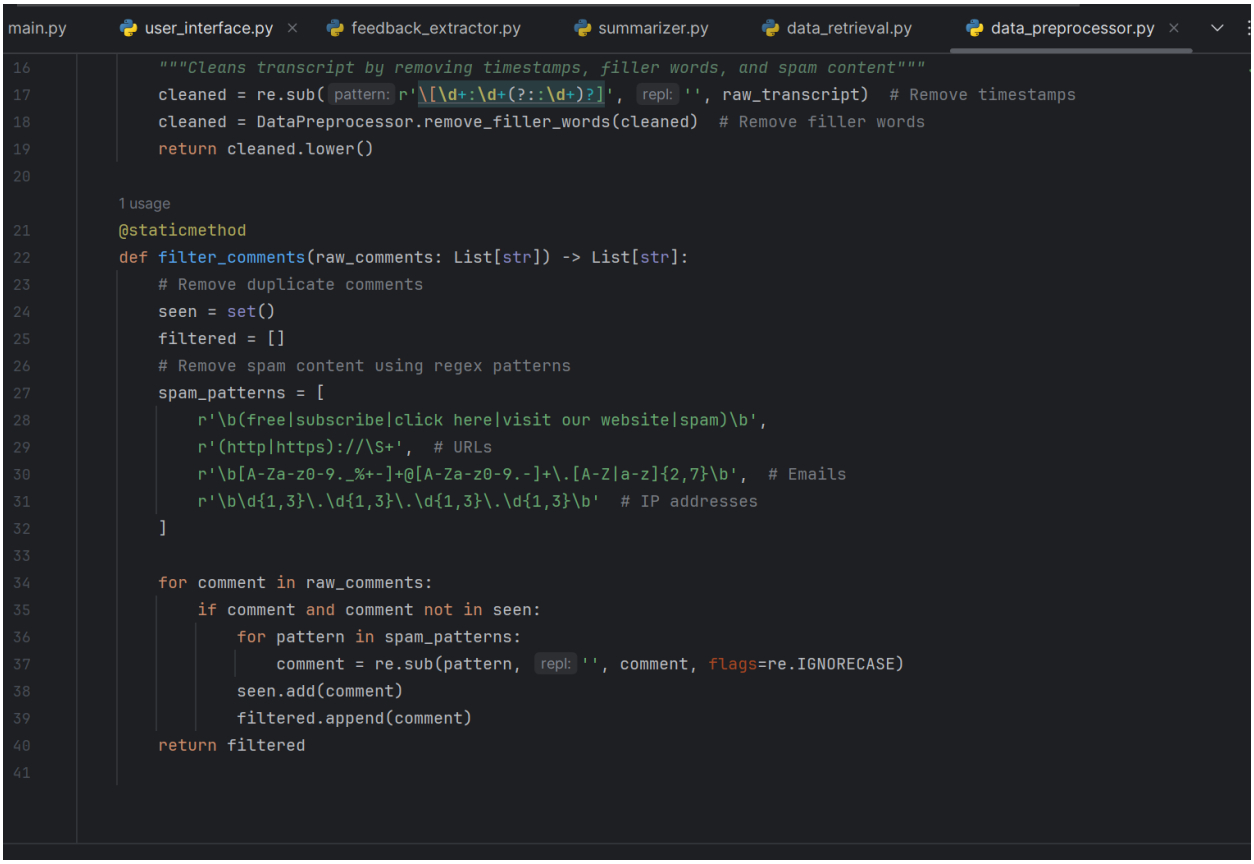


Fig: data_preprocessor.py part 2

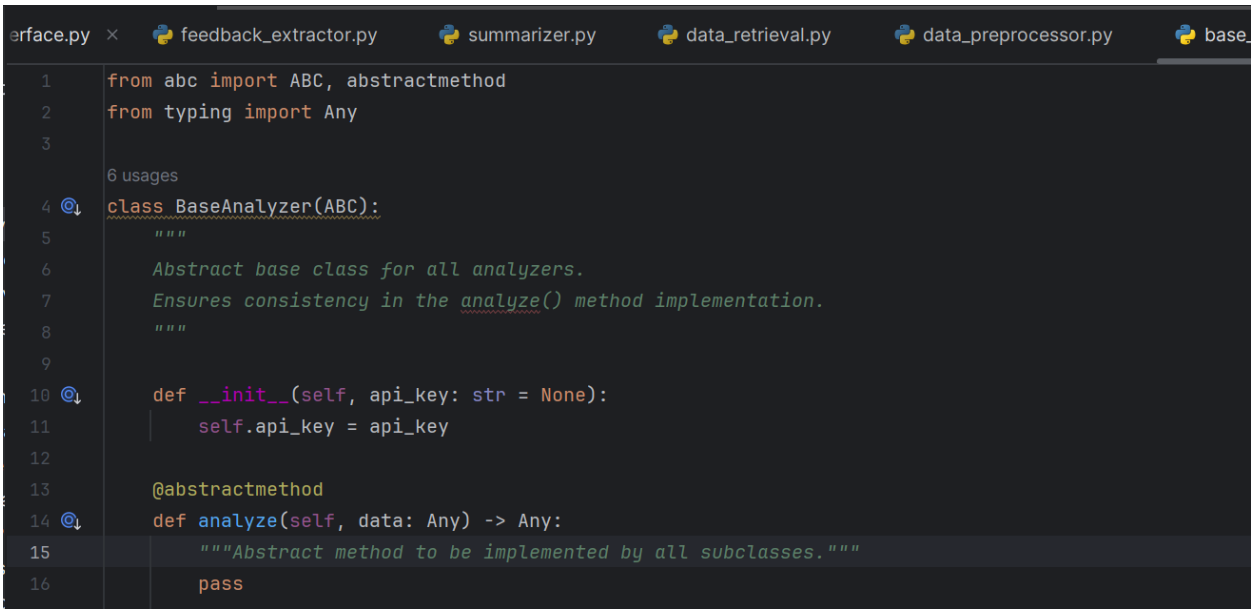


Fig: base analyzer(abstract class connecting summarizer, feedback_extractor and sentiment analysis)

Future deliverables:

- **Move**: Move API keys to a secure configuration file to eliminate hardcoding risks.
- **Optimize**: Optimize comment processing through batching or asynchronous execution.
- **Enhance:** Enhance results UI with sentiment graphs, word clouds, and thumbnails.
- **Transition**: Transition of the application from desktop (Tkinter) to a web-based interface.
- **Improve**: Improve spam filtering using smarter rule sets or lightweight ML classifiers.

Invoice (in hours):

Documentation	Ayush – 33, Anthony – 31, Nabin – 32
Research	Ayush – 19, Anthony – 16, Nabin – 17
Design	Ayush – 16, Anthony – 10, Nabin – 14
Code	Ayush – 18, Anthony – 18, Nabin - 19
Testing	Ayush – 9, Anthony – 10, Nabin – 7

Estimated for Remaining Work:

As per Product Backlog: 50 - 20 = 30 days.

5 extra days to complete what didn't work well in Sprint 2.

Predicted Total: 35 days

Actual Total: 30 days

Tools Used:

- **Software used for this system – PyCharm**
- **Software used to produce your documentation – Microsoft Word and PowerPoint**
- **Other Tools – https://www.draw.io/**

1. Quality Attributes – Runtime Qualities

Attribute	Bus Rank	Design align	Run time Quality Attribute Documents
Performance			**Definition:** System performance describes attributes of the system including navigational abilities and connectivity with interfacing systems. May include system response time, number of transactions per minute, and other critical performance characteristics which define this project. **Reason for Business Rank <> 3:** Performance is essential to deliver timely video summaries and real-time sentiment analysis for a responsive user experience. **Explanation of Design Alignment <> Meets:** The system employs optimized API calls, efficient data retrieval, and parallel processing to ensure performance targets are met.
Reliability			**Definition:** Reliability defines the ability of the system to operate correctly over time. Includes consistent stability of the system during available hours.

			Reason for Business Rank <> 3: High reliability is critical to maintain user trust and ensure continuous availability of video data and analysis. **Explanation of Design Alignment <> Meets:** Redundant mechanisms, robust error handling, and comprehensive logging are implemented to achieve the required reliability.
Scalability			**Definition:** Scalability defines the parameters of growth within a given period. The solution must scale so that additional capacity can be added expediently in a technically and economically feasible way. **Reason for Business Rank <> 3:** Scalability is needed to handle increasing volumes of video and comment data as the user base grows. **Explanation of Design Alignment <> Meets:** A modular and distributed architecture enables the system to scale horizontally as needed.
Security			**Definition:** Security is the ability of the system to resist unauthorized attempts to access the system and/or its components. **Reason for Business Rank <> 3:** Security protects sensitive video metadata, transcripts, and user data from unauthorized access. **Explanation of Design Alignment <> Meets:** The system employs encryption, access controls, and secure API integrations to meet stringent security requirements.
Availability			**Definition:** Availability defines the days and hours the system is anticipated and/or required to be operational. **Reason for Business Rank <> 3:** Continuous system availability is crucial to support global users and uninterrupted service. **Explanation of Design Alignment <> Meets:** The

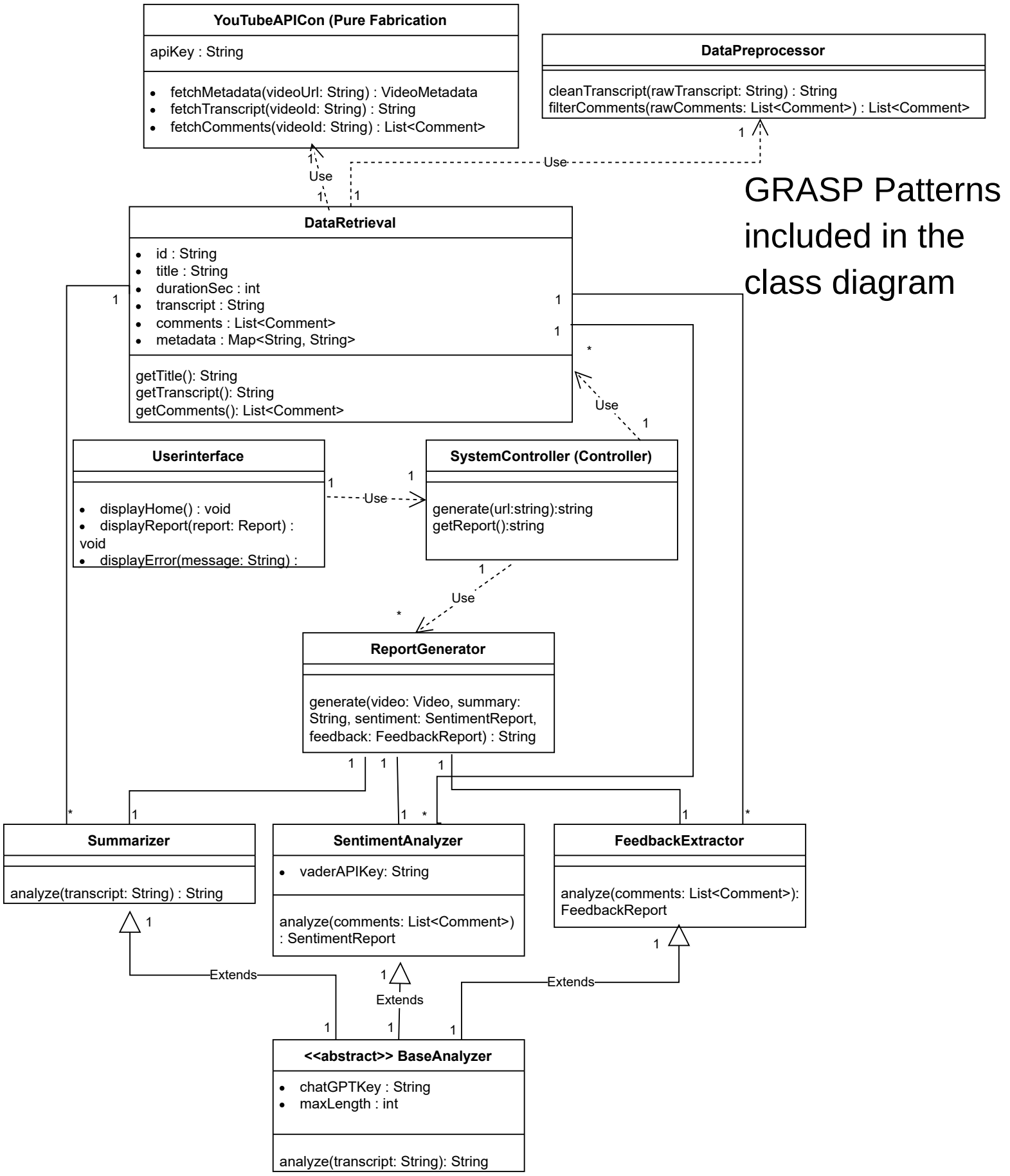
			deployment includes redundant systems and failover mechanisms to ensure 24/7 availability.
Supportability			**Definition:** Supportability defines the requirement that problems can be diagnosed and addressed quickly, with appropriately trained resources. **Reason for Business Rank <> 3:** Quick diagnosis and resolution of issues minimize downtime and maintain service quality. **Explanation of Design Alignment <> Meets:** Integrated logging, monitoring tools, and automated alerts are in place to support rapid troubleshooting.
Traceability			**Definition:** Traceability defines the ability to trace the history, application, or location of an item or activity by means of recorded identification. **Reason for Business Rank <> 3:** It is important for audit trails and debugging to have full traceability of data and changes. **Explanation of Design Alignment <> Meets:** Detailed logging of data transformations (e.g., transcript cleaning, comment filtering) ensures complete traceability throughout the processing pipeline.
User Usability			**Definition:** User Usability defines how easy the interface is to use to successfully complete tasks, including clear navigation, terminology, and error messaging. **Reason for Business Rank <> 3:** A user-friendly interface directly impacts user satisfaction and adoption. **Explanation of Design Alignment <> Meets:** The design features an intuitive dashboard with clear visualizations for video summaries, sentiment analysis, and structured feedback.

2. Quality Attributes- Development Qualities

Attributes	Bus Rank	Design Align	Run Time Quality Attributes Documents
Reusability			**Definition:** The reusability of the system is the ability to reuse portions of the system in other applications. **Reason for Business Rank <> 3:** High reusability reduces development time and cost by allowing components to be leveraged in future projects. **Explanation of Design Alignment <> Meets:** The system is designed with modular components (data retrieval, transcript cleaning, sentiment analysis) that can be reused across different applications.
Extensibility			**Definition:** Extensibility defines the degree to which the system can be extended to incorporate new functionality without changing the fundamental architecture, design, or core source code. **Reason for Business Rank <> 3:** Extensibility is essential to accommodate evolving business needs and integration with additional data sources. **Explanation of Design Alignment <> Meets:** Published interfaces such as APIs and metadata definitions allow for seamless extension of system capabilities.
Modifiability			**Definition:** Modifiability (maintainability) defines the efficiency of making enhancements or maintenance changes. **Reason for Business Rank <> 3:** High modifiability minimizes disruption and cost when updating the system. **Explanation of Design Alignment <> Meets:** A well-structured, documented, and modular codebase allows for quick modifications without affecting core functionality.
Ease of Integration			**Definition:** Ease of Integration defines how easily the application or its components can be integrated or packaged

			with new applications. **Reason for Business Rank <> 3:** Seamless integration is key to interoperability with external systems and tools. **Explanation of Design Alignment <> Meets:** The system uses standardized, RESTful APIs and common data formats to facilitate integration.
Phase-In "Ability"			**Definition:** Phase In "ability" (subset-ability) defines how the application accommodates implementation/rollout schedules without impacting overall design. **Reason for Business Rank <> 3:** Phased deployment minimizes risk and allows for early user feedback. **Explanation of Design Alignment <> Meets:** The architecture supports incremental deployment, enabling features to be released in stages.
Conceptual Integrity			Definition: Conceptual Integrity refers to the ability of the architecture to communicate a clear, concise vision in line with the enterprise architecture. Reason for Business Rank <> 3: Ensures a unified approach and consistency across the system, which is vital for long-term maintainability. Explanation of Design Alignment <> Meets: Consistent design principles and a unified processing pipeline ensure all components work cohesively.
Testability			**Definition:** Testability describes the ability to test each component independently (unit tests) as well as in an integrated environment. **Reason for Business Rank <> 3:** High testability reduces risk of defects and ensures reliable system performance with each update. **Explanation of Design Alignment <> Meets:** A comprehensive suite of automated tests is integrated into the

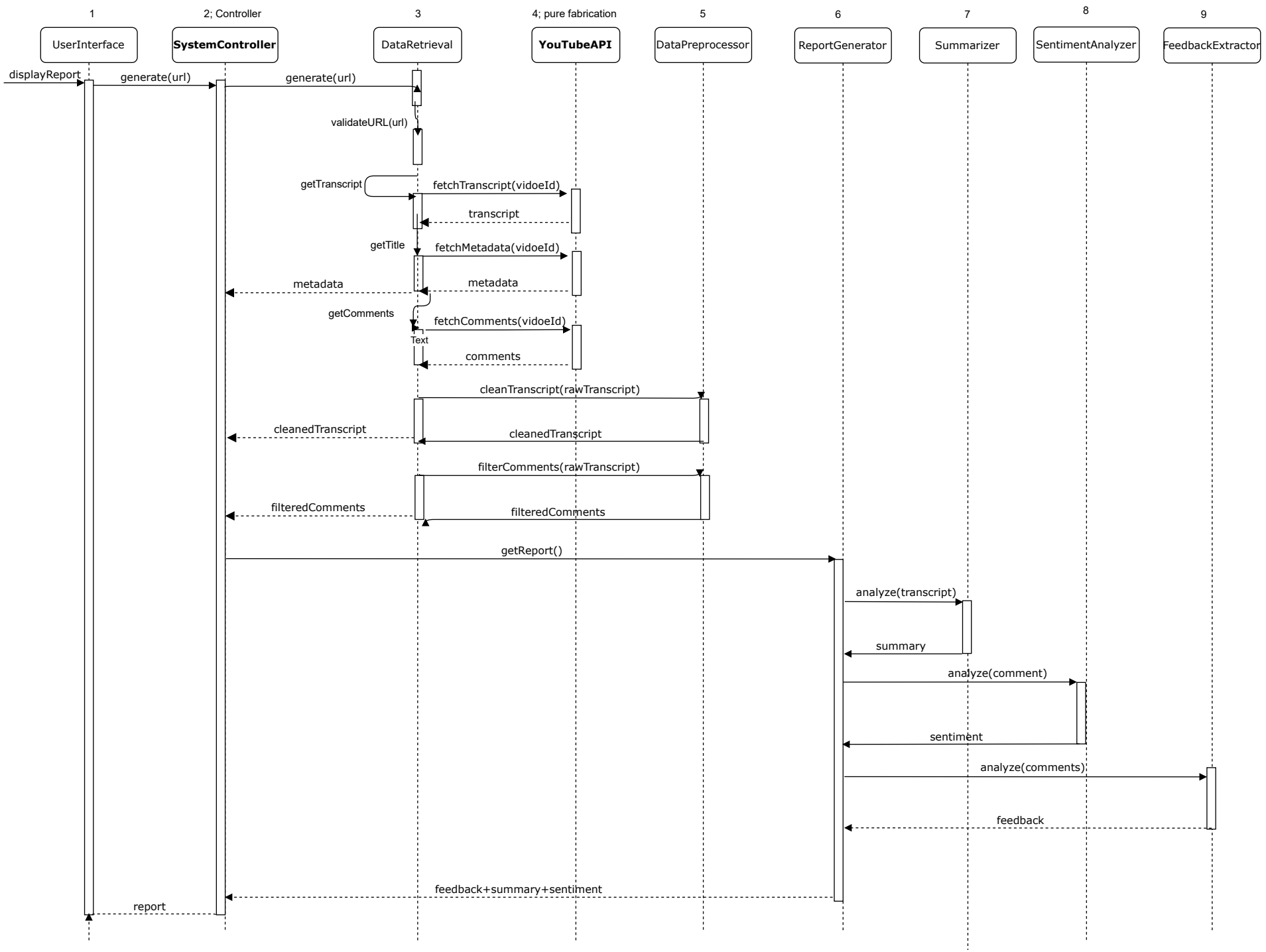
			CI/CD pipeline to verify all critical components.
Portability			**Definition:** Portability measures the ease with which the system can be moved to different platforms (hardware, operating systems, virtualization, etc.). **Reason for Business Rank <> 3:** Portability provides flexibility in deployment and minimizes vendor lock-in. **Explanation of Design Alignment <> Meets:** The system is developed using cross-platform frameworks and containerization, ensuring compatibility across environments.
Developer Complexity			**Definition:** Developer Complexity is how easy it is for developers to understand and create the system. **Reason for Business Rank <> 3:** Lower complexity accelerates onboarding and reduces development errors. **Explanation of Design Alignment <> Meets:** The architecture is simplified with clear module boundaries and detailed documentation, reducing the overall complexity.
Buildability			**Definition:** Buildability defines the degree to which the system can be compiled, packaged, archived, and deployed using automated tools and standard processes. **Reason for Business Rank <> 3:** Efficient build processes accelerate time-to-market and ensure consistency. **Explanation of Design Alignment <> Meets:** The integration of an automated CI/CD pipeline and build automation tools streamlines deployment and minimizes errors.

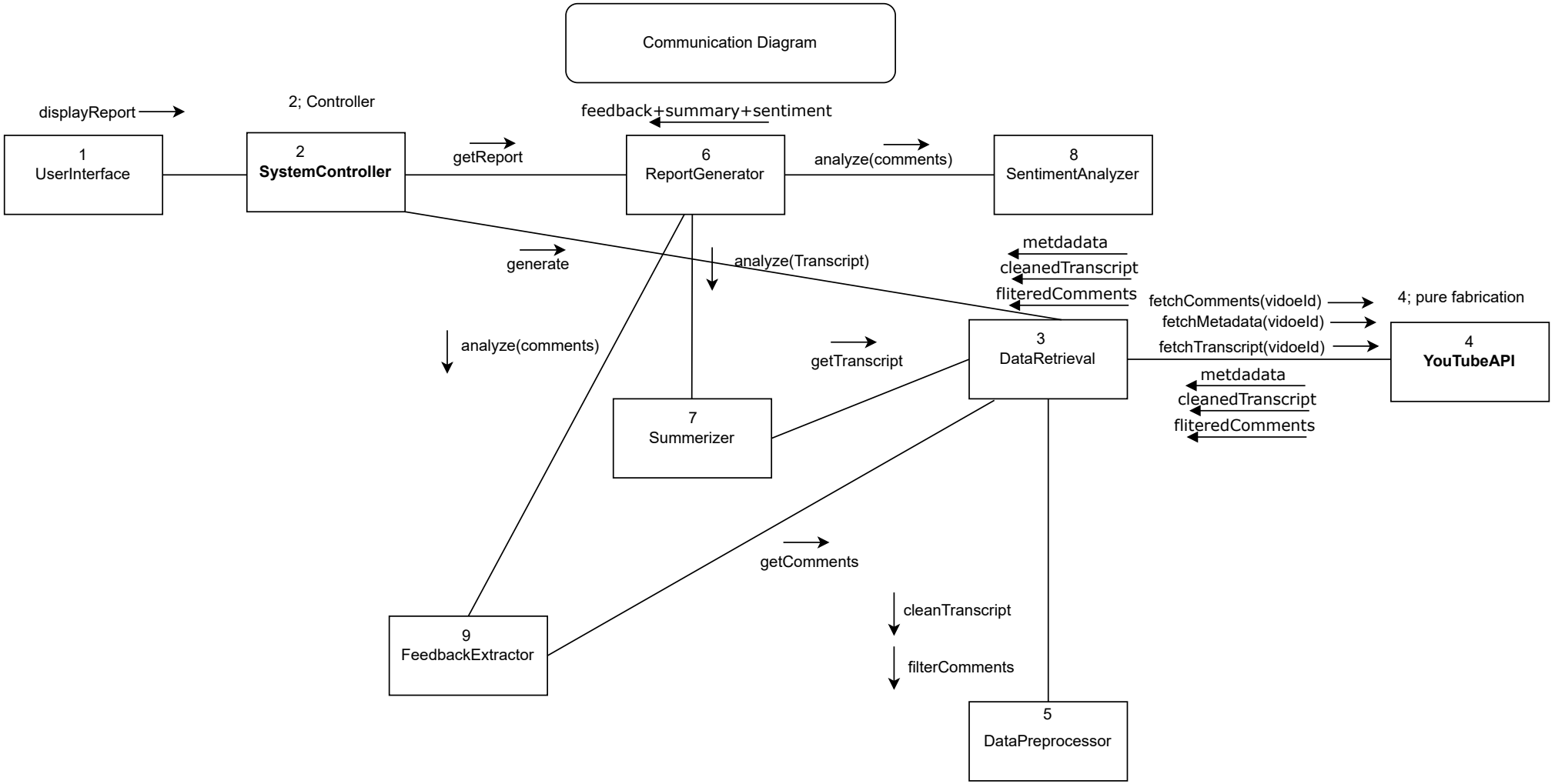


GRASP Patterns included in the class diagram

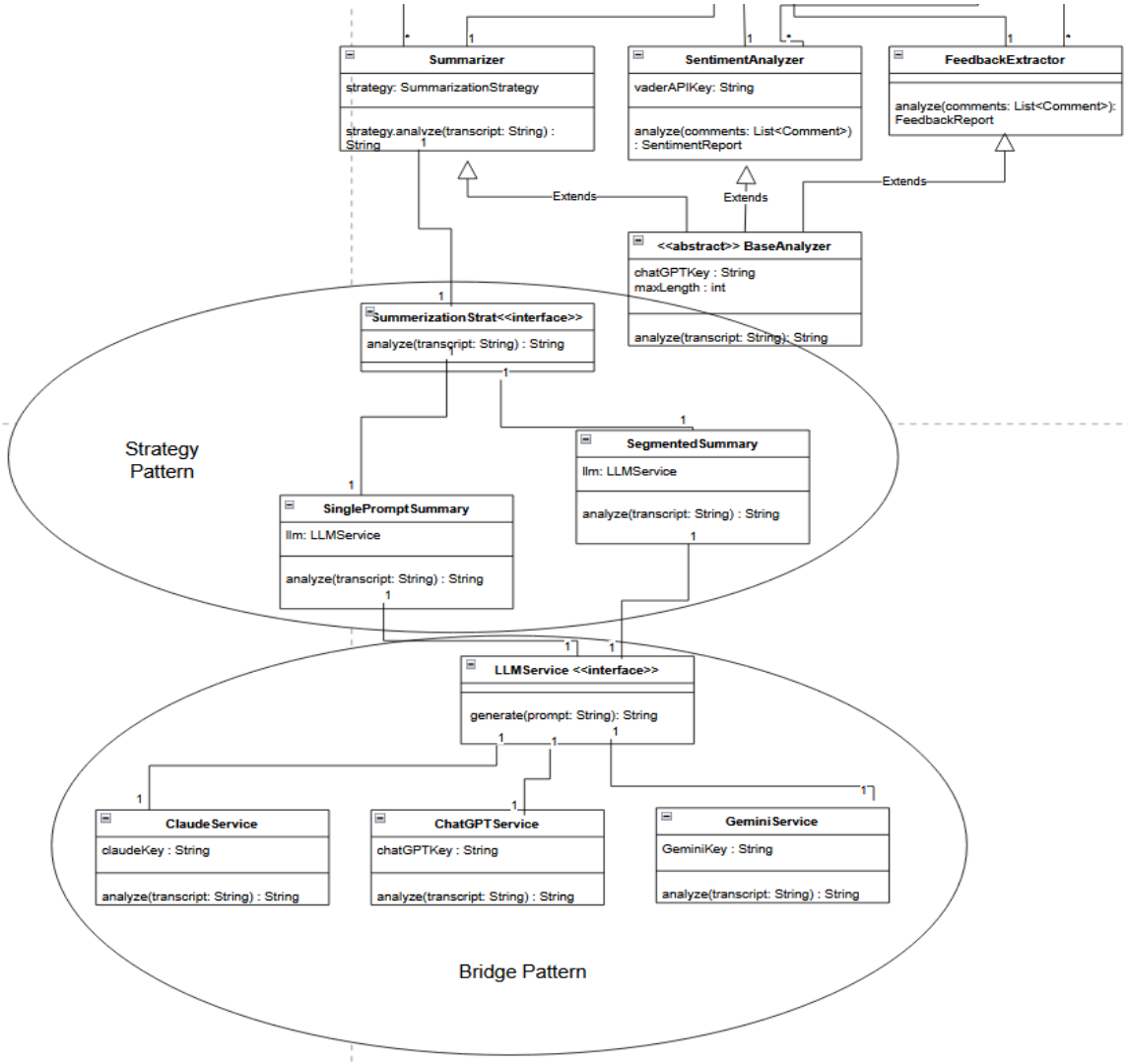
Sequence Diagram

GRASP Patterns: Sequence Diagram



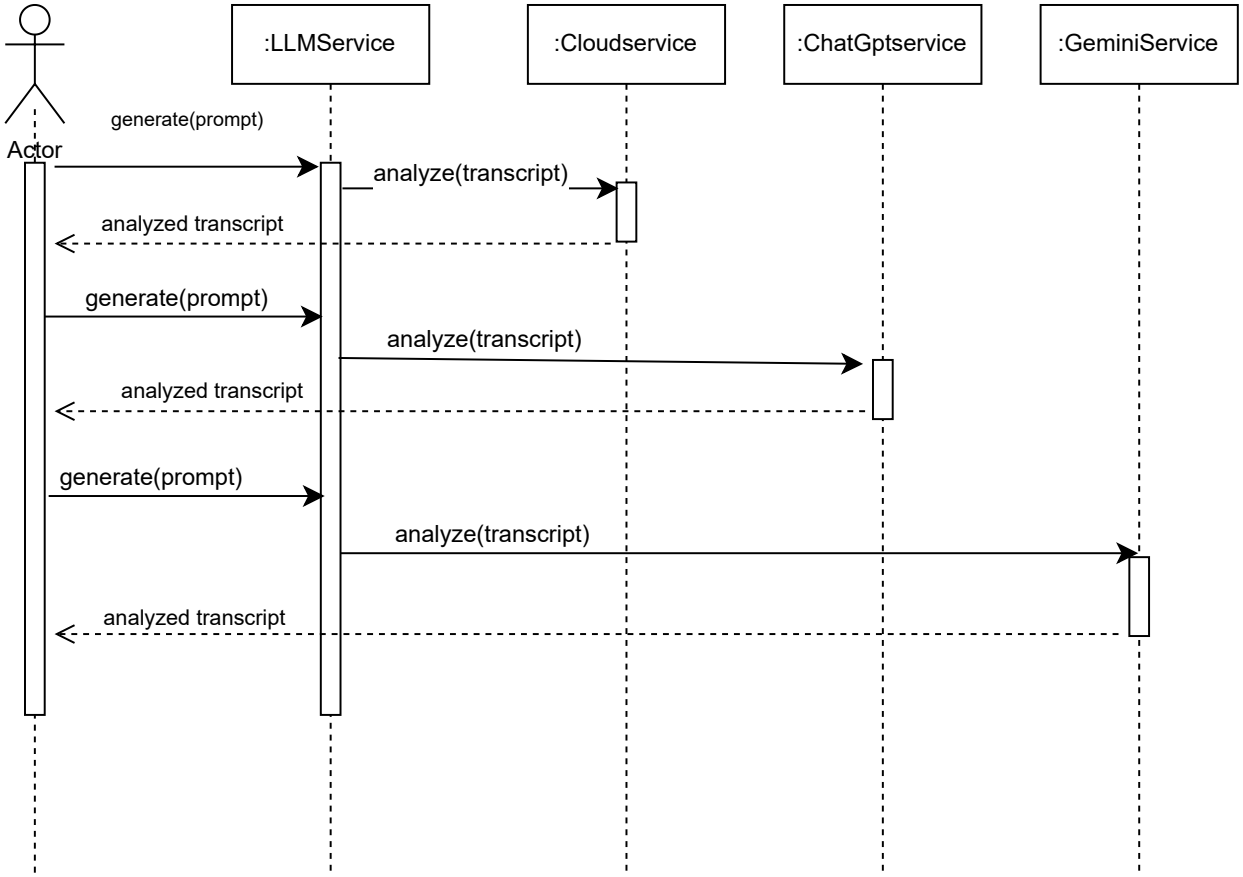


GRASP Patterns: Communication Diagram



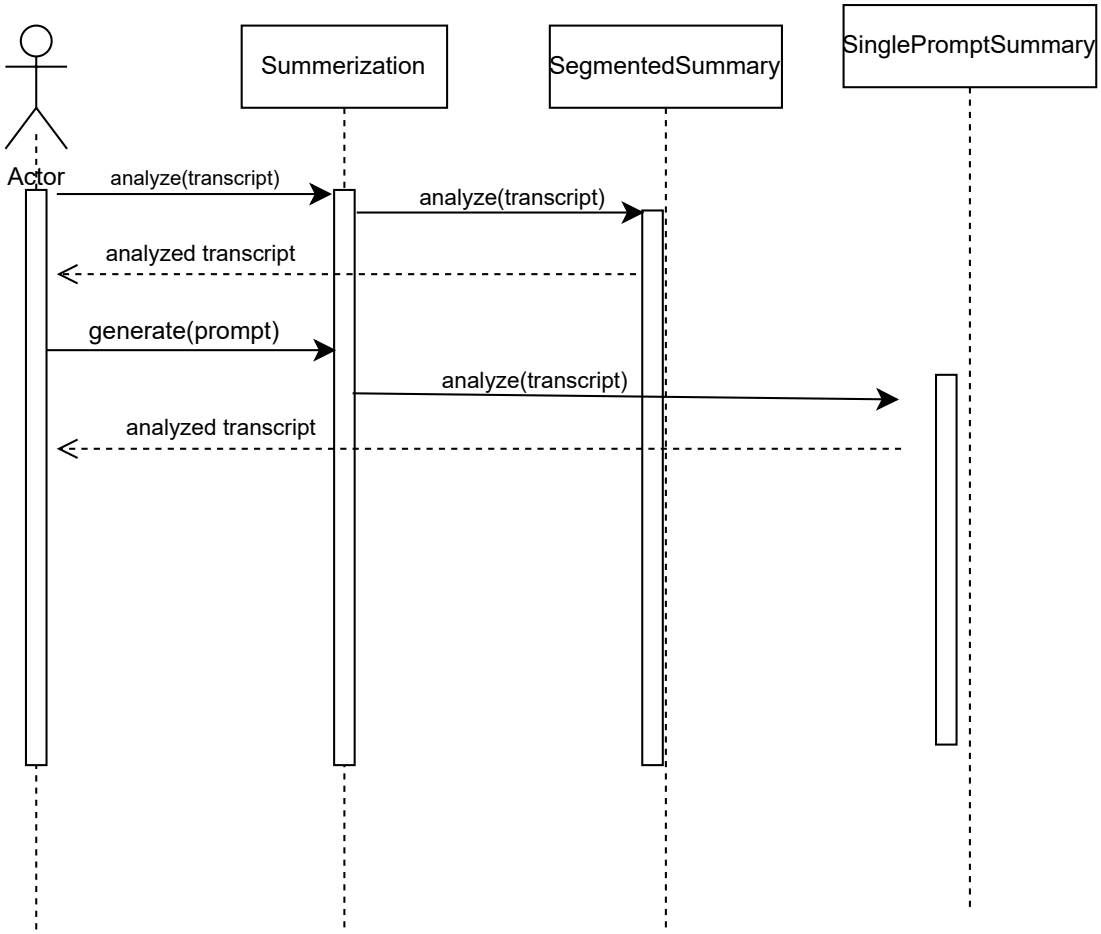
DIAGRAM

GOF PATTERNS: Strategy and Bridge Patterns



GOF Patterns:

Bridge pattern



GOF Patterns:

Strategy pattern