FOOTBALL HIGHLIGHTS GENERATOR

PROJECT REPORT

Submitted By

BIYAS MOHAMMED M
ASI20CA023

CLEETO ITTIACHAN
ASI20CA024

SHAREEH BIN SHAKKIR
ASI20CA055

Under the guidance of

Prof. SHYAMA R

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

CSE (Artificial Intelligence)

ADI SHANKARA INSTITUTE OF ENGINEERING & TECHNOLOGY

AUGUST 2023

CERTIFICATE

Certified that this is a bonafide record of the miniproject

entitled **"FOOTBALL HIGHLIGHTS GENERATOR"**

Submitted by

BIYAS MOHAMMED M
ASI20CA023

CLEETO ITTIACHAN
ASI20CA024

SHAREEH BIN SHAKKIR
ASI20CA055

during the year 2022-23 in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology in CSE (Artificial Intelligence)

Internal Guide | External Supervisor Project

Coordinator | Head of the Department

ACKNOWLEDGEMENT

We have taken a lot of effort to build this project. However, completing this project would not have been possible without the support and guidance of a lot of individuals. We would like to extend our sincere thanks to our Principal **Dr. Sreepriya S** , **Prof. P.V. Rajaraman** Head of Department, Computer Science And Engineering And Mini Project Coordinators **Prof. T Sobha** and **Prof. Vydehi S**. We are highly indebted to **Prof. Shyama R** for her guidance and supervision. We would like to thank her for providing the necessary information and resources for this project. We would like to express our gratitude towards all teaching and non-teaching staff of AI & CSE Department , parents ,friends , well wishers and Almighty for their kind cooperation and encouragement which help us a lot in completing this project. Our thanks and appreciations also go to our colleagues in developing the project. Thank you to all the people who have willingly helped us out with their abilities.

ABSTRACT

The "Sports Video Summary Generator" is a graphical user interface (GUI) application designed to facilitate the automated summarization of sports videos. The application employs various techniques from audio and video processing to identify and extract key moments from sports videos, creating concise and engaging highlights. The user interface offers functionalities for video upload, summary length selection, and video playback. The generated video summaries capture significant events within the chosen duration, providing an efficient way for viewers to relive the excitement of a game without having to watch the entire video.

The application is built using the Python programming language and leverages libraries such as `tkinter` for GUI creation, `librosa` for audio analysis, `numpy` for numerical calculations, `pandas` for data management, `matplotlib` for data visualization, and `moviepy` for video processing. Users can upload sports videos in popular formats (e.g., .mp4, .avi, .mkv) and specify their desired summary length. The application then extracts audio features, analyzes energy patterns, and identifies segments of interest. It creates a summary video by concatenating subclips corresponding to these key moments, resulting in an engaging highlights reel.

The GUI provides clear and intuitive controls for uploading videos, selecting summary lengths, generating highlights, and playing the generated summary video. The program encapsulates audio and video processing complexities within an easy-to-use interface, enabling sports enthusiasts to efficiently create personalized video summaries that capture the most thrilling moments of a game.The "Sports Video Summary Generator" serves as a demonstration of how technology can enhance the sports viewing experience by automating the process of creating condensed and captivating video highlights.

CONTENTS

LIST OF FIGURES

CHAPTER1
INTRODUCTION

In the realm of sports entertainment, the power of technology has significantly transformed the way audiences engage with their favorite games. With the rapid evolution of digital media, the demand for efficient and immersive experiences has led to the development of novel solutions aimed at enhancing sports consumption. One such innovation is the "Football Highlights Generator," a software application designed to revolutionize the way football enthusiasts relive the excitement of matches.

The "Football Highlights Generator" is a user-friendly graphical user interface (GUI) application that employs cutting-edge audio and video processing techniques to create condensed and captivating video highlights from football matches. This application caters to the desires of modern sports fans who seek to savor the most exhilarating moments of a football game without investing the time required to watch the entire match. By enabling users to curate their own highlight reels, this tool bridges the gap between traditional full-length viewing and the need for easily digestible content.

This project leverages a range of sophisticated libraries, including `tkinter` for GUI development, `librosa` for audio analysis, `numpy` for numerical computations, `pandas` for data management, `matplotlib` for data visualization, and `moviepy` for video editing. Users can effortlessly upload football match videos, select the desired summary length, and obtain a dynamically edited highlights reel that encapsulates the game's most crucial moments.

In this report, we delve into the design, implementation, and functionality of the "Football Highlights Generator." We explore the underlying algorithms that process audio energy patterns to identify significant segments of the match. The application’s intuitive interface guides users through the video upload, summary length selection, and highlight generation processes. By offering a seamless bridge between technology and sports entertainment, the "Football Highlights Generator" showcases the potential of leveraging innovation to provide immersive and tailored experiences for football enthusiasts.

The subsequent sections of this report detail the various components of the "Football Highlights Generator," its algorithmic approach to creating highlights, and the benefits it offers to both users and the broader sports community. The project embodies a synergy of technology and passion for sports, culminating in a solution that transforms the way we relive and celebrate the essence of football matches.

CHAPTER 2
LITERATURE SURVEY

The concept of summarizing sports events using audio and video processing techniques has gained significant traction in recent years. Researchers and developers alike have explored ways to efficiently extract key moments from lengthy sports broadcasts to cater to modern viewers' limited attention spans. Several studies and projects have contributed to the evolution of sports video summarization and highlight generation.

1. Automatic Video Summarization: Research by Zhou et al. introduced an approach to automatically summarizing soccer games using a combination of video and audio features. The work highlighted the importance of detecting highlight-worthy moments based on audio-visual cues and demonstrated how intelligent algorithms can create compelling summaries.
2. Dynamic Video Summarization: The work of Wei et al. focused on real-time summarization of sports videos. The proposed system considered both game dynamics and user preferences to generate personalized highlights in real-time. This approach acknowledged the need to provide viewers with a tailor-made experience.
3. Audio Analysis for Highlight Detection: In the domain of audio analysis, Nwe et al. explored audio-based techniques to identify important segments in soccer matches. The research demonstrated the potential of audio features, such as crowd reactions and commentator excitement, to detect critical moments.
4. Interactive Video Summarization: A project by Kuang et al. introduced an interactive sports video summarization tool that allowed users to adjust summary parameters on-the-fly. This approach empowered viewers to customize their highlights and aligns with the user-centric design of modern applications.
5. Video Editing and User Interaction: The "SportsMash" project by Garg and Murthy proposed a system for interactive and automatic summarization of cricket matches. The tool integrated video editing techniques with user interactions to facilitate the creation of dynamic highlights tailored to individual preferences.
6. Machine Learning for Sports Video Analysis:Survey literature on machine learning and computer vision approaches applied to sports video analysis. Explore how these techniques are employed to detect key events, players, and highlight-worthy moments in football matches.
7. Automatic Sports Highlight Generation: Study research on automatic highlight generation techniques for sports videos. Analyze the algorithms and methodologies used to extract significant segments and create coherent highlights from longer footage.

The "Football Highlights Generator" aligns with these trends in the field of sports video summarization. By leveraging audio and video analysis, it offers an accessible platform for users to effortlessly curate their own football match highlights. The project acknowledges the significance of user interaction and real-time adaptability while addressing the growing demand for personalized and concise sports content consumption.

As technology continues to evolve, the intersection of audio-visual analysis and user-centric design will likely play a pivotal role in shaping the future of sports video summarization. The "Football Highlights Generator" contributes to this evolution by providing an efficient and immersive way for football enthusiasts to re-experience the thrill of matches without compromising on the essence of the game.

CHAPTER 3

SYSTEM DESIGN

3.1 FUNCTIONAL REQUIREMENTS

1. Video Upload :
   - Allow users to upload football match videos in popular formats (e.g., .mp4, .avi, .mkv).
   - Display the selected video's path on the user interface.

2. Summary Length Selection:
   - Provide users with pre-defined summary length options (e.g., 10 seconds, 15 seconds, 20 seconds, 30 seconds).
   - Allow users to select their desired summary length from the available options.

3. Highlight Generation:
   - Analyze audio energy patterns from the uploaded video to identify exciting moments.
   - Compute an energy threshold based on the average energy level.
   - Detect audio segments surpassing the energy threshold as potential highlights.
   - Merge consecutive highlights segments for smoother transitions.

4. Video Subclip Extraction:
   - Extract video subclips corresponding to the identified highlight intervals.
   - Ensure that subclips capture key moments without compromising context.

5. Subclip Concatenation:
   - Concatenate the extracted subclips to create a cohesive highlights video.
   - Produce a seamless video that effectively captures the essence of the match.

6. User Interface Interaction:
   - Display clear instructions and feedback to guide users through the highlight generation process.
   - Provide visual cues to indicate video upload status, selected summary length, and generated highlights availability.
   - Enable users to trigger the highlight generation process with a "Generate Highlights" button.

7. Play Highlights Video:
   - Allow users to play the generated highlights video within the application interface.
   - Enable users to relive the captivating moments of the football match.

8. Result Display:

- Display the path to the generated highlights video for user reference.
- Show informative messages to inform users about the completion of the highlight generation process.

9. Play Video Controls:

- Provide controls for users to pause, play, rewind, and fast-forward the generated highlights video.

10. Cleanup and Resource Management:

- Ensure proper deletion of temporary files, folders, and audio resources after the highlight generation process.

11. Compatibility and Scalability:

- Design the application to handle varying sizes and qualities of football match videos.
- Optimize algorithms for efficient processing even with longer videos.

3.2 ARCHITECTURAL DIAGRAM

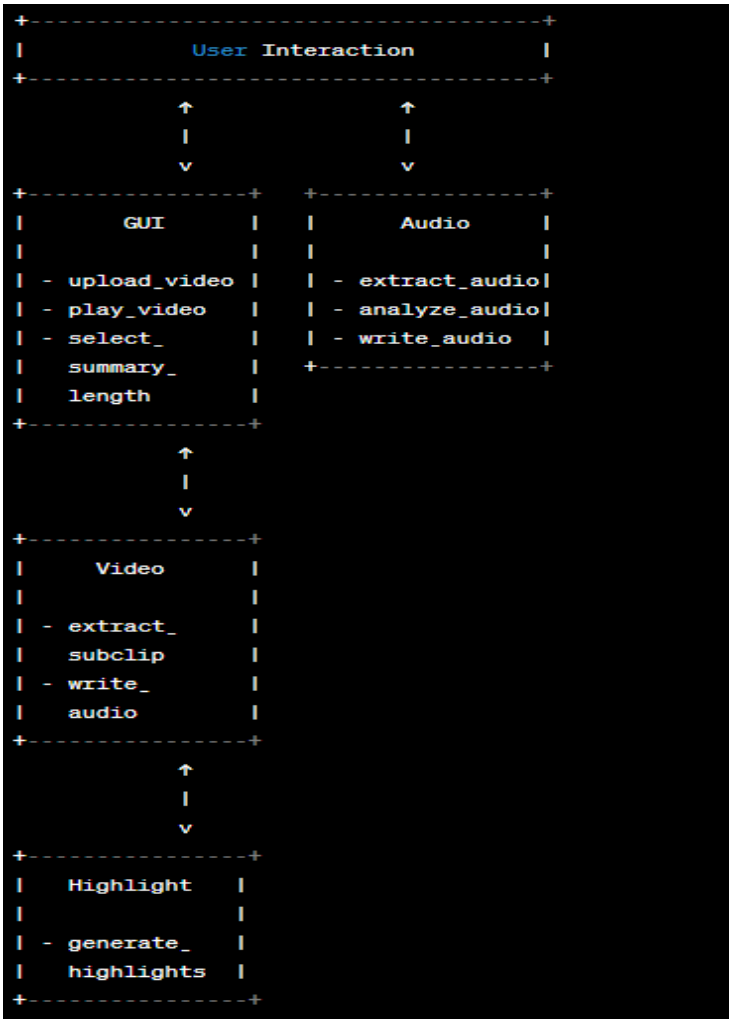


Figure 3.1

3.3 USE CASE DIAGRAM

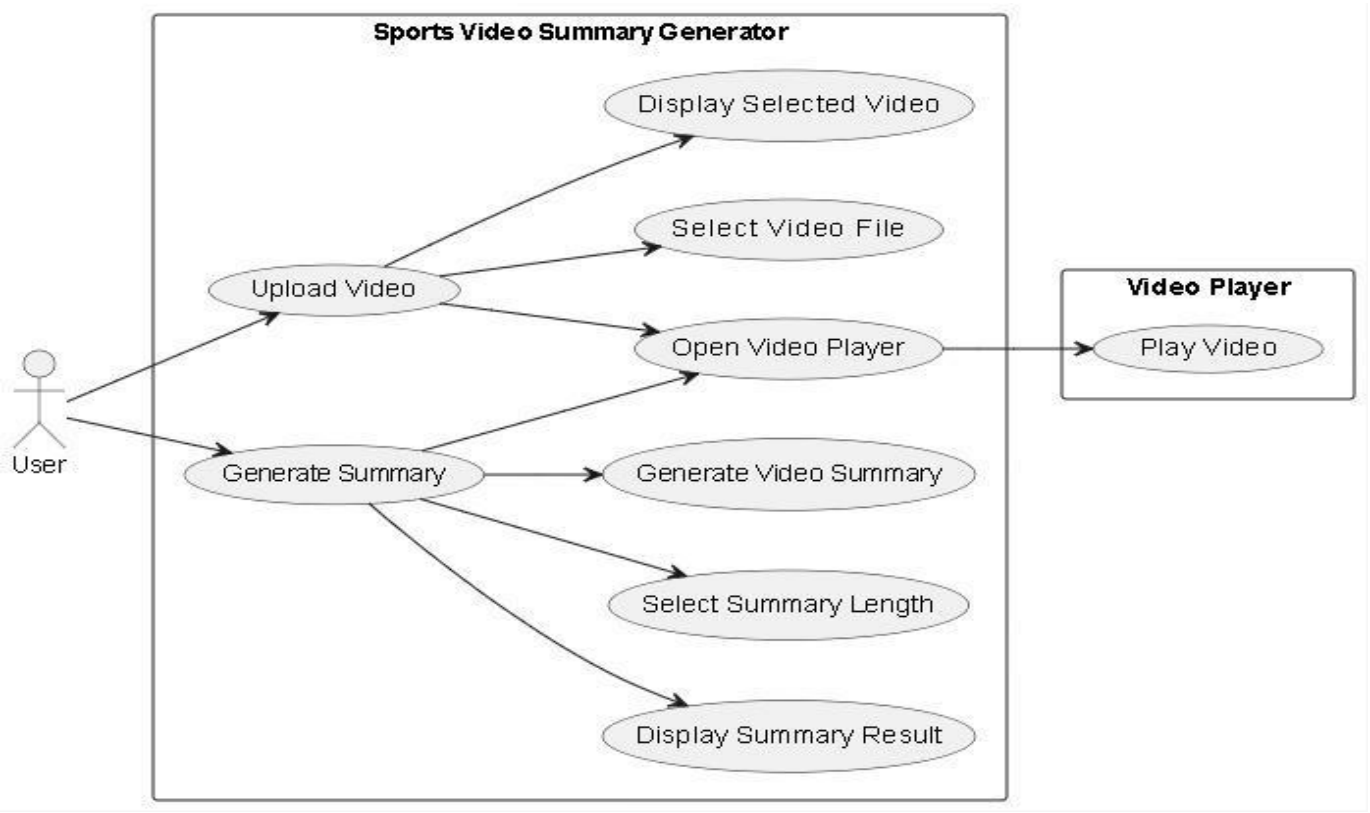


Figure 3.2

3.4 CLASS DIAGRAM

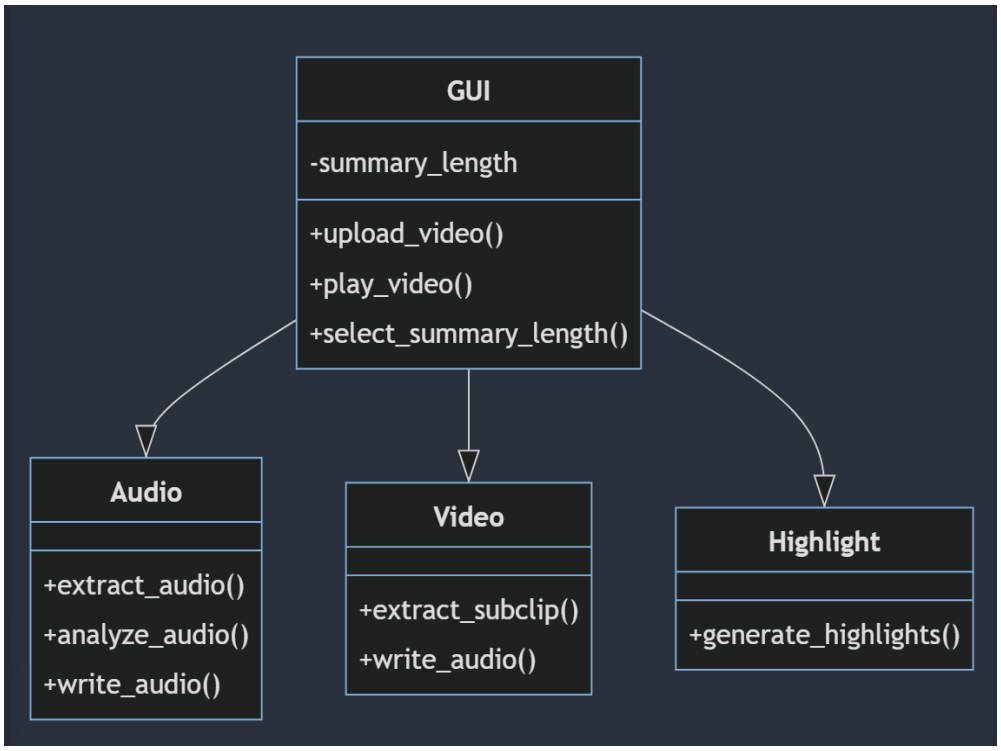


Figure 3.3

3.5 SEQUENCE DIAGRAM

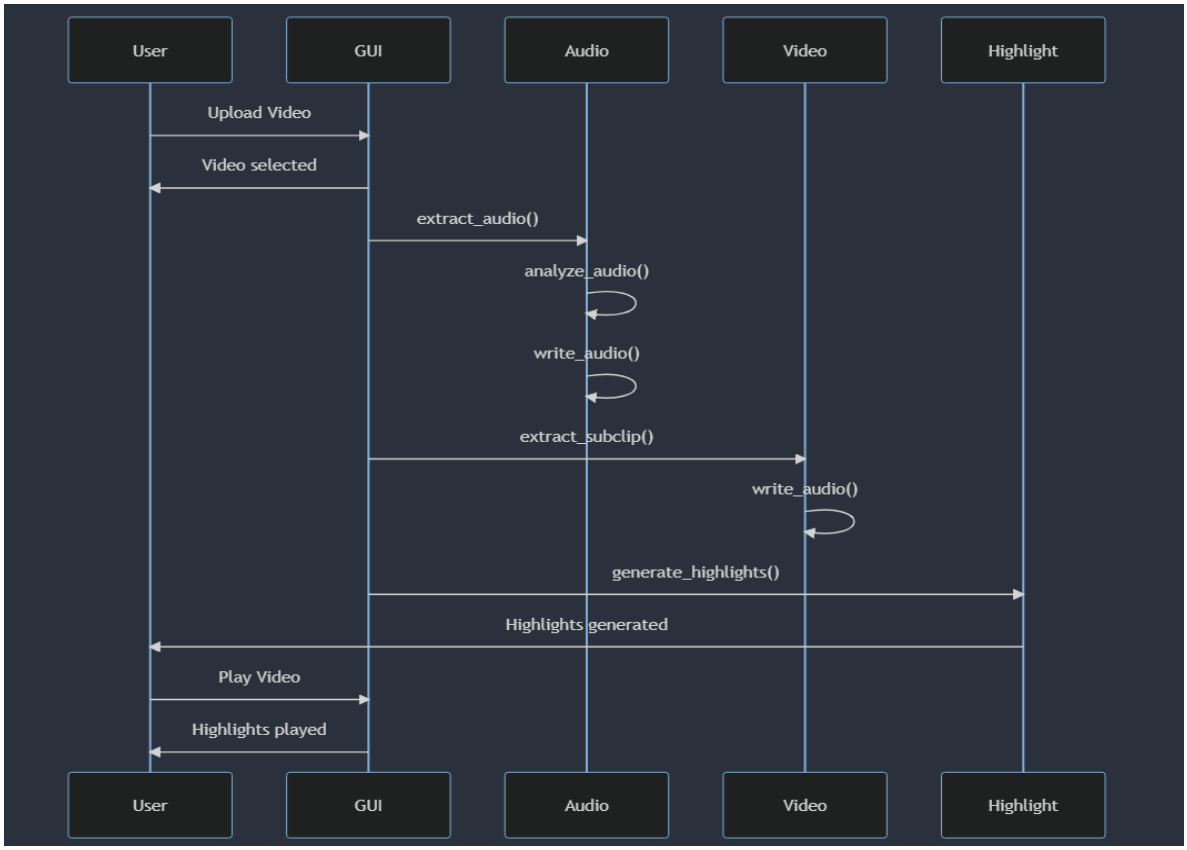


Figure 3.4

3.6 ACTIVITY DIAGRAM

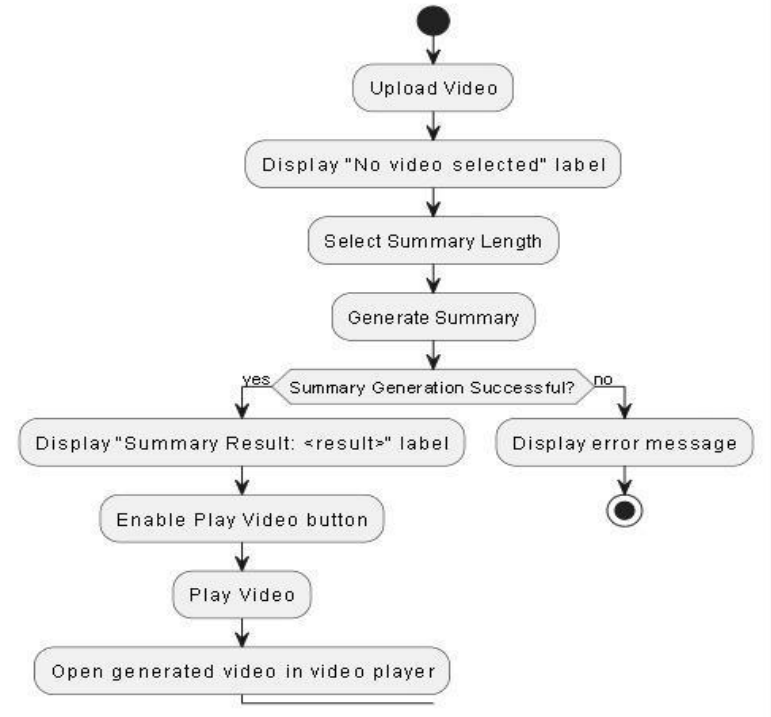


Figure 5

3.7 FUNCTION DESCRIPTIONS

1. Video Selection and Upload:
    - Users can choose their favorite football match video and upload it to the application.
    - This video will serve as the source for creating highlights.
2. Customizable Summary Length:
    - Users have the flexibility to decide how long they want the highlights video to be.
    - They can choose from preset options like 10, 15, 20, or 30 seconds.
3. Automatic Highlights Generation:
    - The application analyzes the uploaded video's audio to identify exciting moments, such as goals or crowd reactions.
    - It then automatically creates a highlights video capturing these moments.
4. Play Highlights Video:
    - Users can instantly play the generated highlights video within the application.
    - This feature allows them to relive the most thrilling segments of the football match.
5. User-Friendly Interface:
    - The application offers a user-friendly graphical interface with intuitive buttons and labels.
    - Users can easily navigate through the process of uploading, generating, and playing highlights.

These functionalities collectively enable users to create personalized football match highlights that encapsulate the excitement of the game in a concise and engaging manner.

3.8 SOFTWARE AND HARDWARE REQUIREMENTS

3.8.1 SOFTWARE REQUIREMENTS

1. Python: The programming language used for developing the application.
2. Tkinter: A Python library for creating graphical user interfaces.
3. tkinterthemes: A Python library for applying themes to Tkinter GUIs.
4. MoviePy: A Python library for video editing and manipulation.
5. Librosa: A Python library for audio analysis.
6. NumPy: A Python library for numerical computations.
7. Pandas: A Python library for data manipulation and analysis.
8. Matplotlib: A Python library for creating visualizations.
9. FFmpeg: A multimedia framework for video and audio processing.

3.8.2 HARDWARE REQUIREMENTS

1. Computer: A computer system capable of running Python and the required libraries.
2. Processor: A modern processor (e.g., Intel Core i5 or equivalent) for efficient processing.

3. Memory (RAM): At least 4 GB of RAM for smooth execution.
4. Storage: Adequate storage space for storing the application, videos, and generated highlights.
5. Display: A display screen with a resolution of 1280x720 or higher for clear GUI visualization.
6. Sound Output: Speakers or headphones for audio playback of highlights.

CHAPTER 4

TEST PLAN & RESULT

4.1: TEST PLAN

1. Upload Video Functionality:
   - Test Scenario: Verify if users can successfully upload a football match video.
   - Test Steps:
     1. Launch the application.
     2. Click the "Upload Video" button.
     3. Select a sample video file.
   - Expected Result: The selected video's path should be displayed on the GUI.

2. Summary Length Selection:
   - Test Scenario: Check if users can choose their preferred summary length.
   - Test Steps:
     1. Launch the application.
     2. Choose a different summary length (e.g., 15 seconds).
   - Expected Result: The selected summary length should be displayed in the dropdown menu.

3. Automatic Highlights Generation:
   - Test Scenario: Validate if the application can generate highlights automatically.
   - Test Steps:
     1. Upload a sample football match video.
     2. Choose a summary length (e.g., 10 seconds).
     3. Click the "Generate" button.
   - Expected Result: The application should generate a highlights video based on the chosen length.

4. Play Highlights Video:
   - Test Scenario: Ensure users can play the generated highlights video.
   - Test Steps:
     1. Upload a sample football match video.
     2. Choose a summary length (e.g., 20 seconds).
     3. Click the "Generate" button.
     4. Click the "Play Video" button.
   - Expected Result: The generated highlights video should play within the application.

5. User Interface Interaction:

- Test Scenario: Verify the responsiveness of the GUI.
- Test Steps:
  1. Upload a sample football match video.
  2. Choose a summary length.
  3. Click the "Generate" button.
  4. Observe GUI updates during the process.
- Expected Result: The GUI should provide real-time feedback and updates to guide users through each step.

5. Performance Testing:

- Test Scenario: Assess the application's performance with different video sizes.
- Test Steps:
  1. Upload a short video (e.g., 1 minute).
  2. Upload a longer video (e.g., 10 minutes).
- Expected Result: The application should handle both short and longer videos without significant slowdowns.

6. Usability Testing:

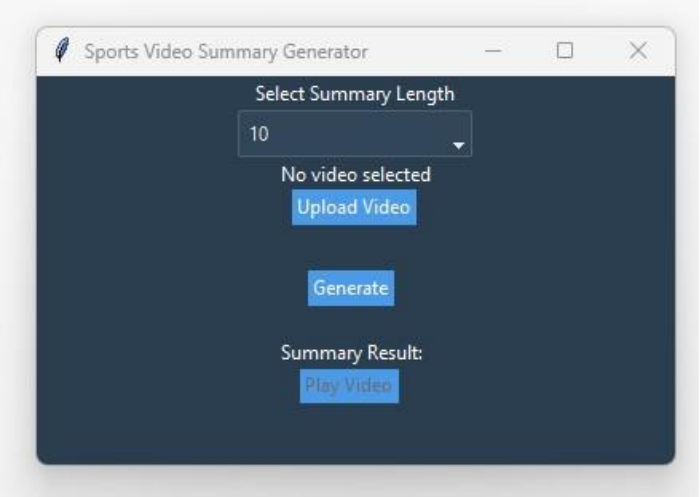
- Test Scenario: Evaluate the application's usability from a user's perspective.
- Test Steps:
  1. Involve users with varying levels of technical expertise.
  2. Ask them to upload a video, choose a summary length, generate highlights, and play the video.
- Expected Result: Users should be able to perform these actions easily and without confusion.

7. Cleanup and Resource Management:

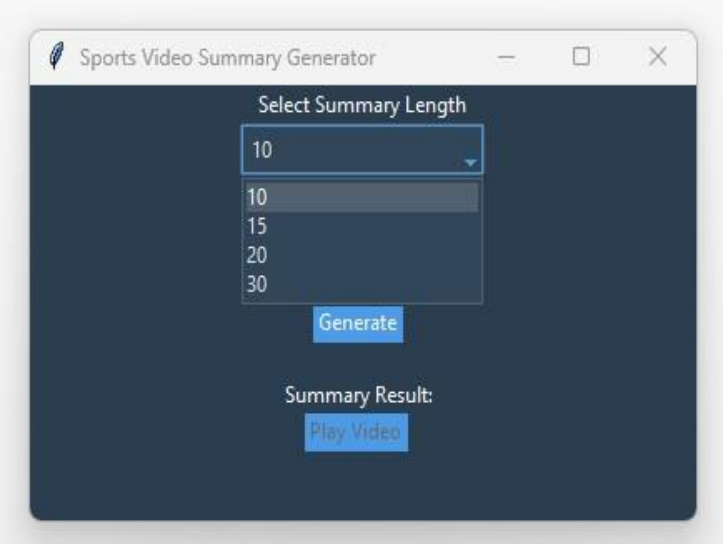
- Test Scenario: Verify if temporary files and resources are properly managed.
- Test Steps:
  1. Upload a video, generate highlights, and play the video.
  2. Check for leftover temporary files/folders.
- Expected Result: The application should delete temporary files and folders after generating and playing highlights.

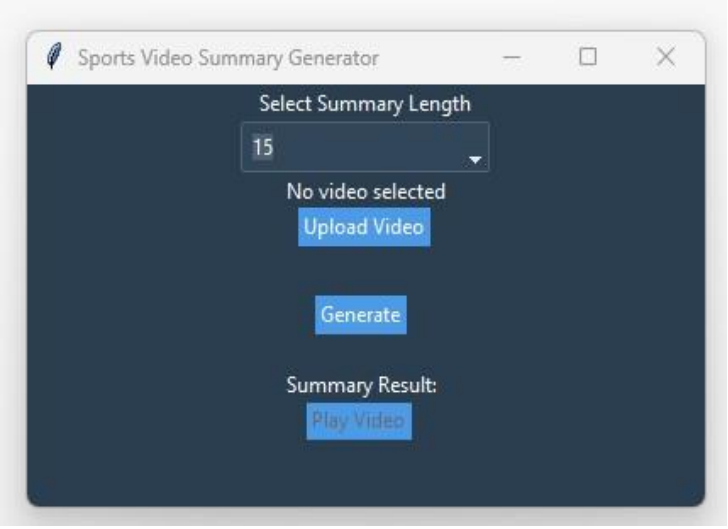
4.2: RESULT

Application GUI:

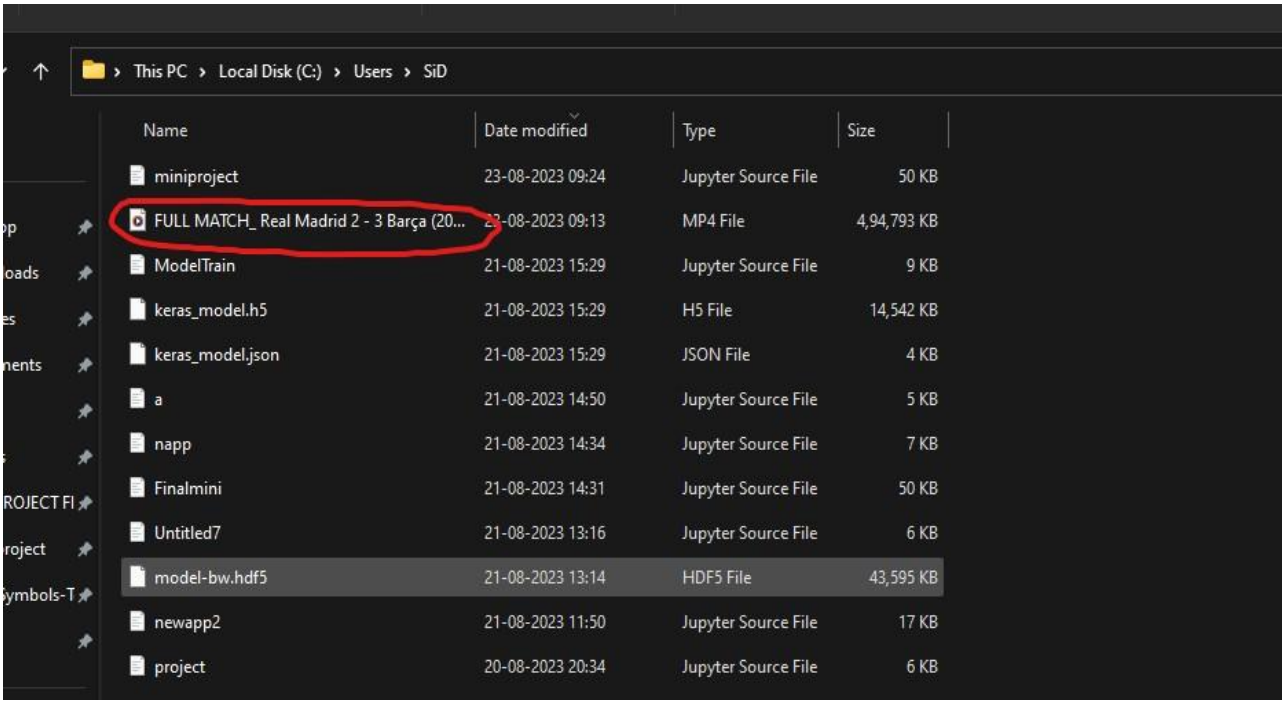


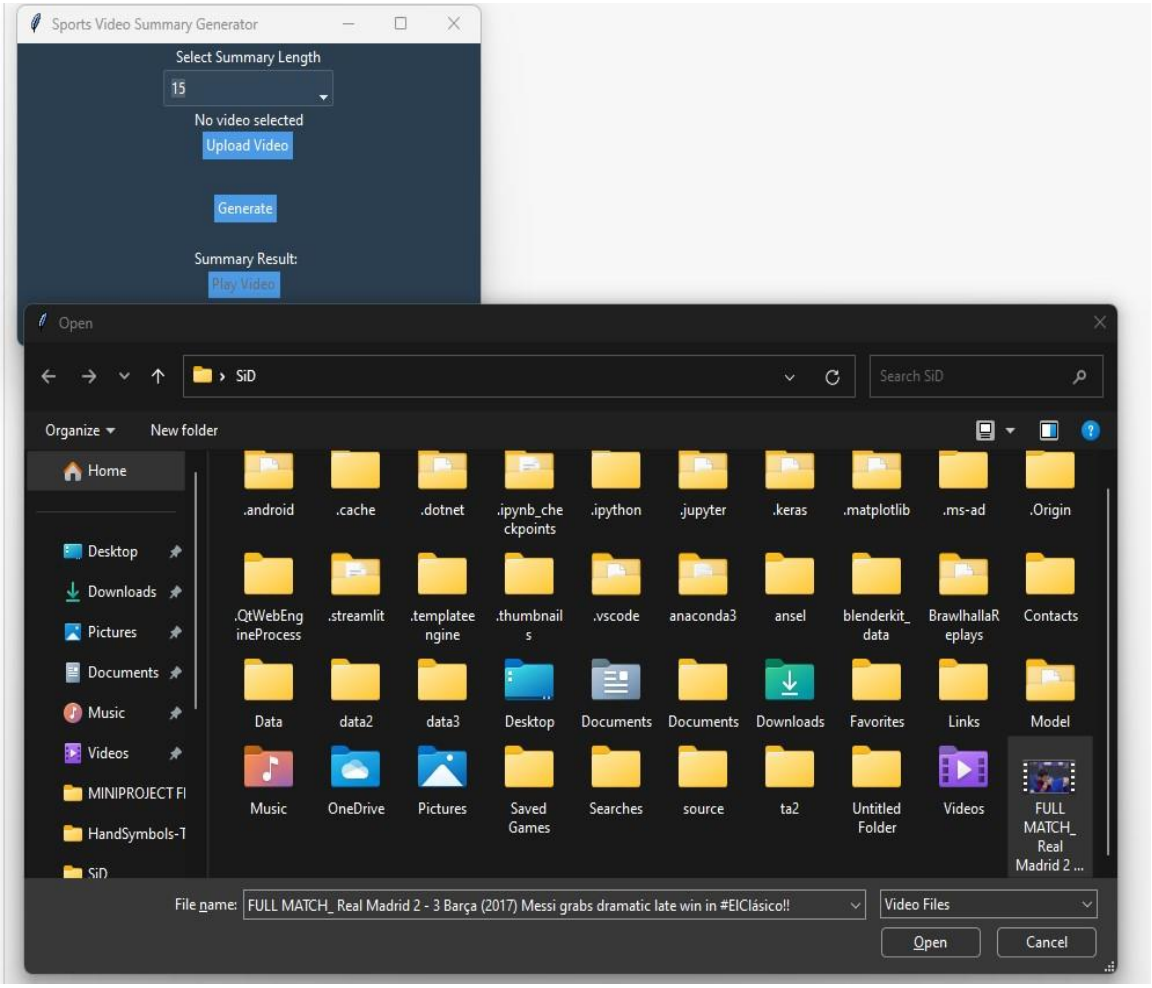
Selecting length:



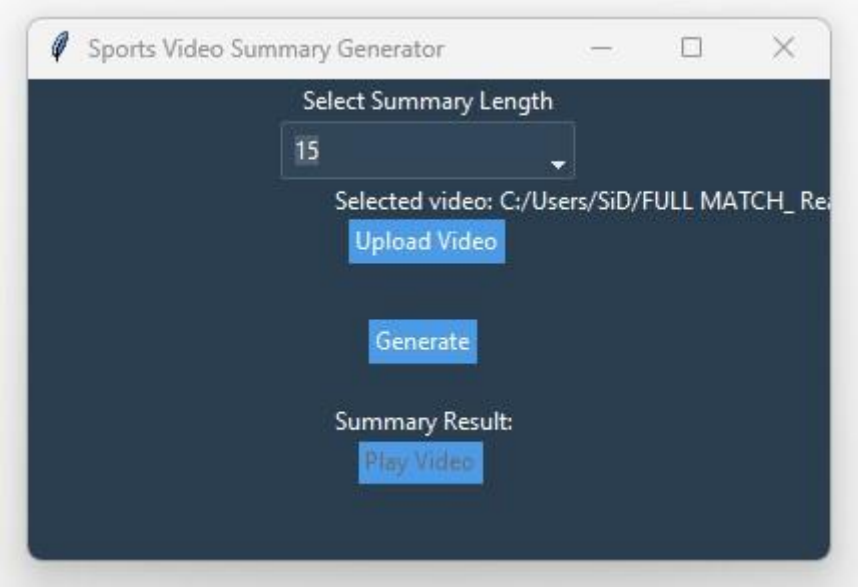


Selecting video:



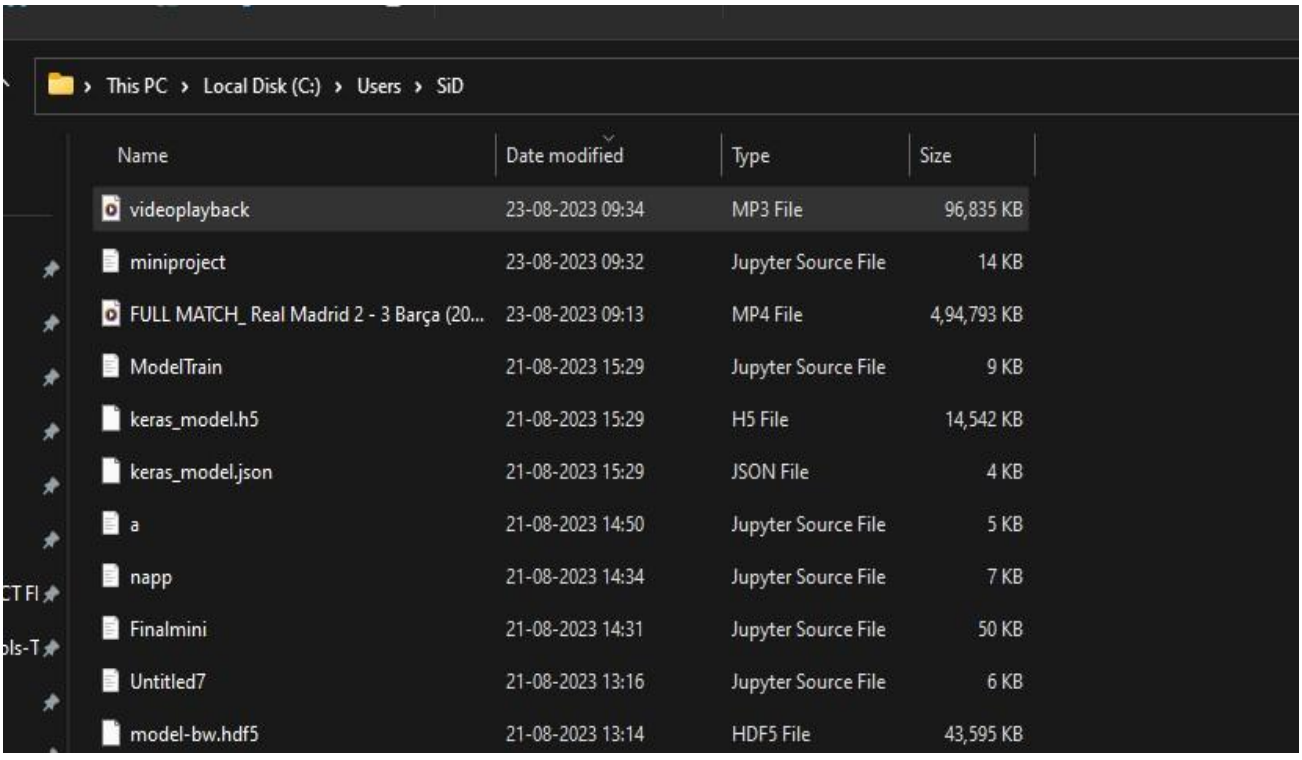


Video selected and then we will click Generate, the Play video button is disabled:

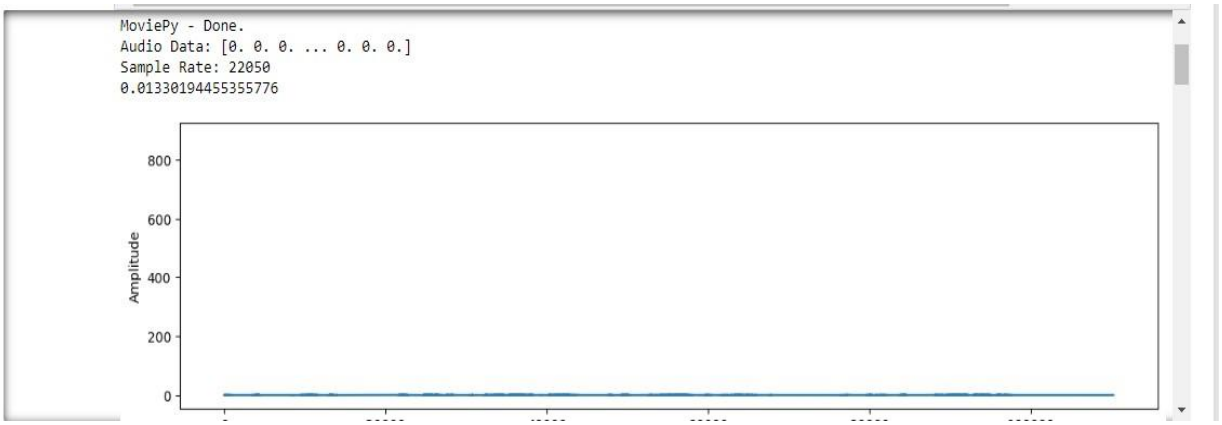


The python program begins creats temporary audio file of the video:

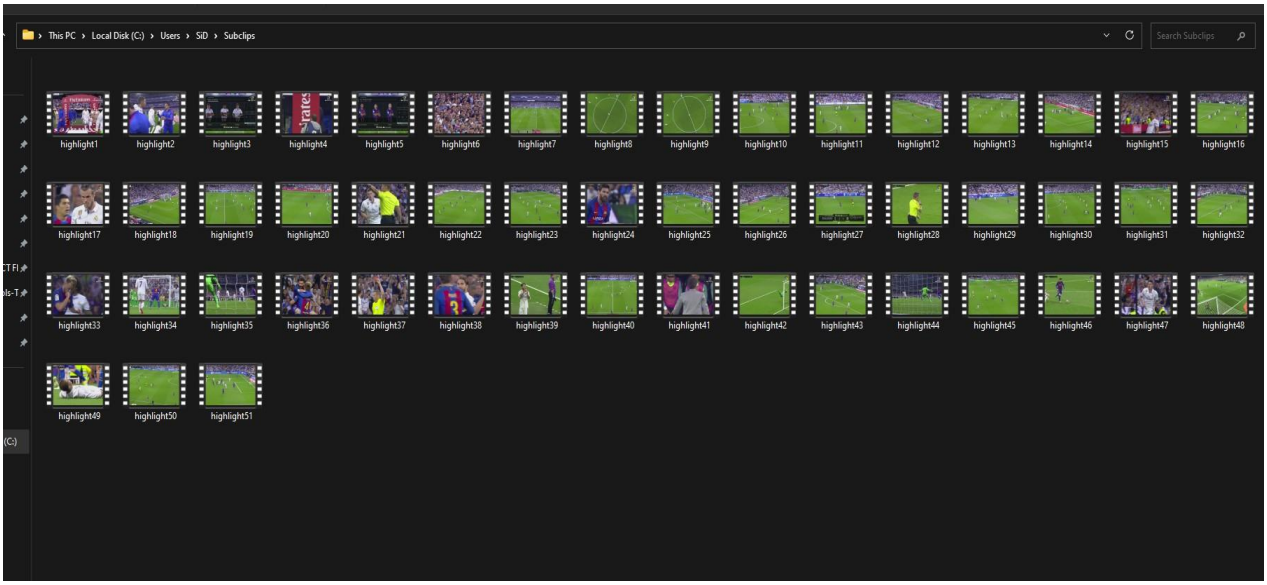
```
Selected video file: C:/Users/SiD/FULL MATCH_ Real Madrid 2 - 3 Barça (2017) Messi grabs dramatic late win in #ElClásico!!.mp4
option 15
MoviePy - Writing audio in videoplayback.mp3
chunk:  24%|████████████                                  | 32686/136653 [00:14<00:50, 2075.31it/s, now=None]
```



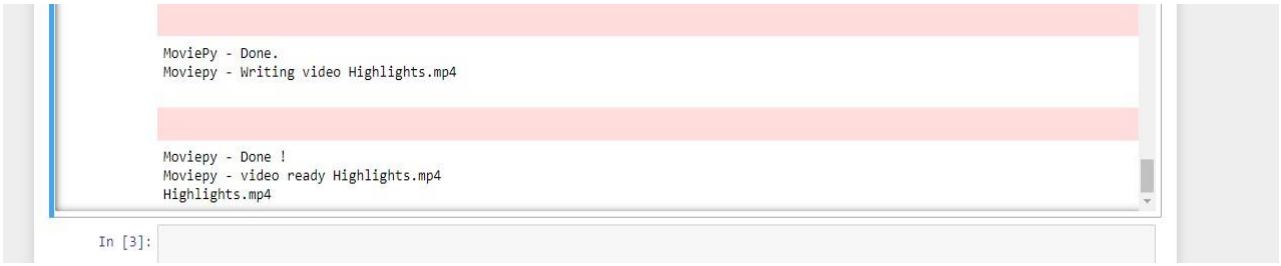
Audio sampling in program:

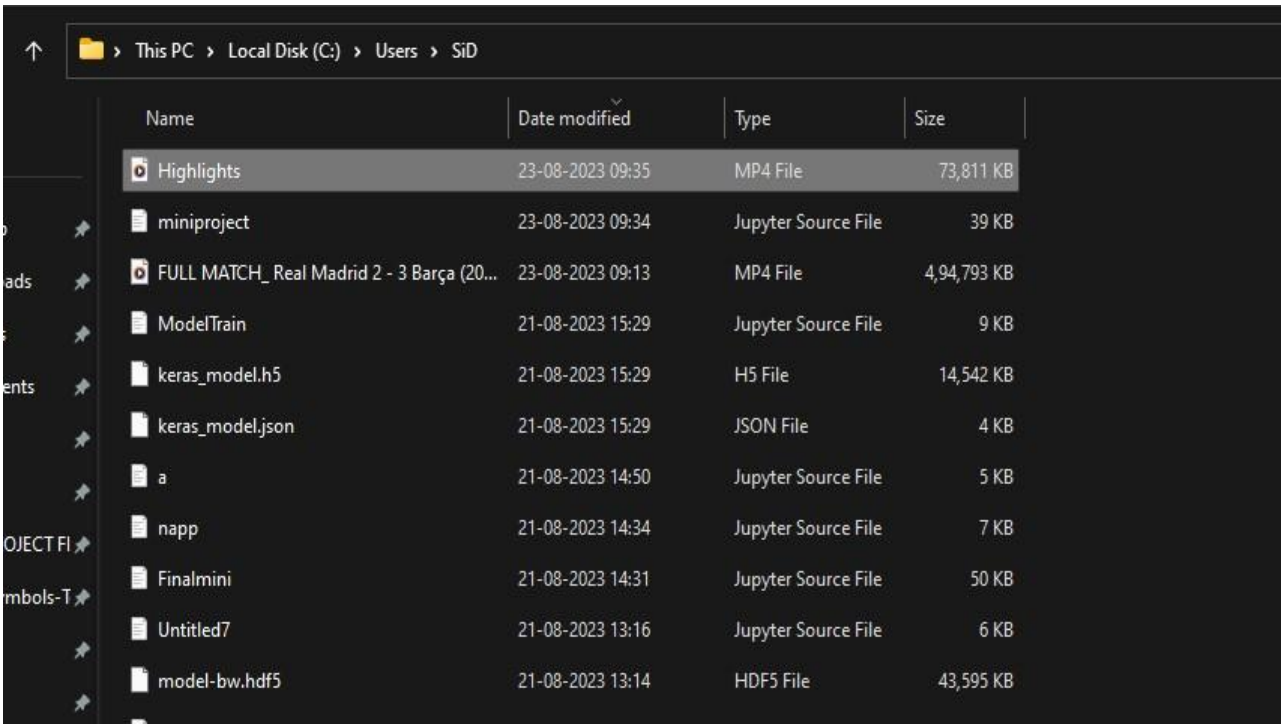


Subclips being extracted:

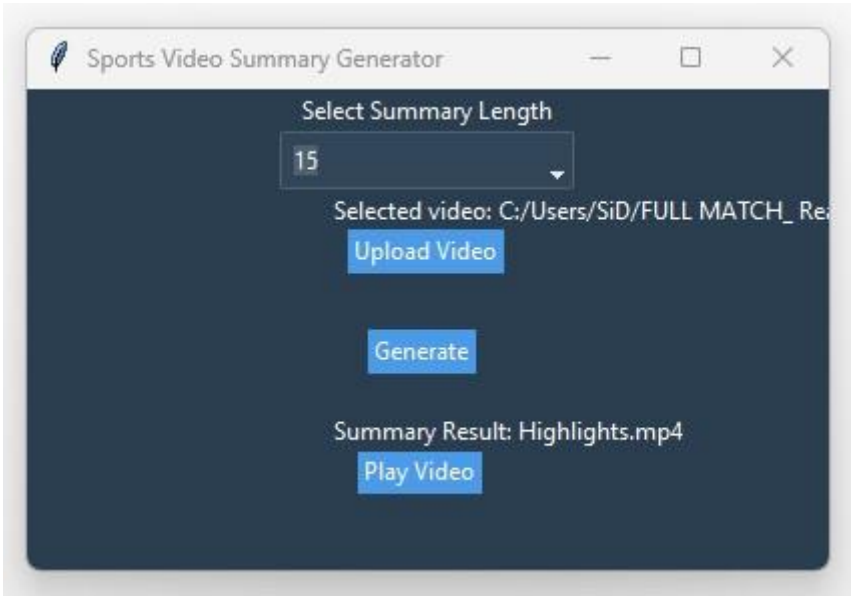


The final output file is created:



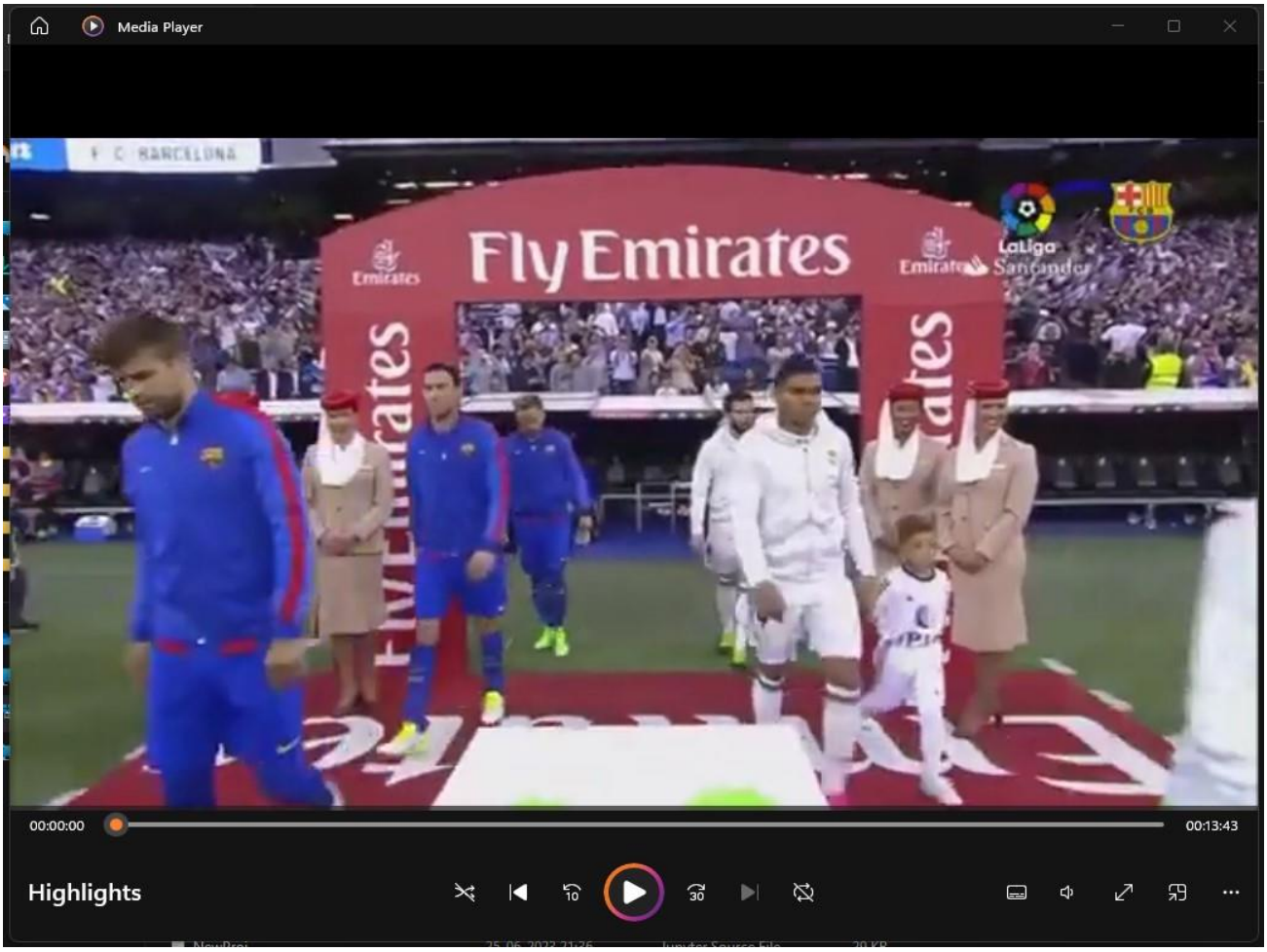


The Play video button gets enabled:



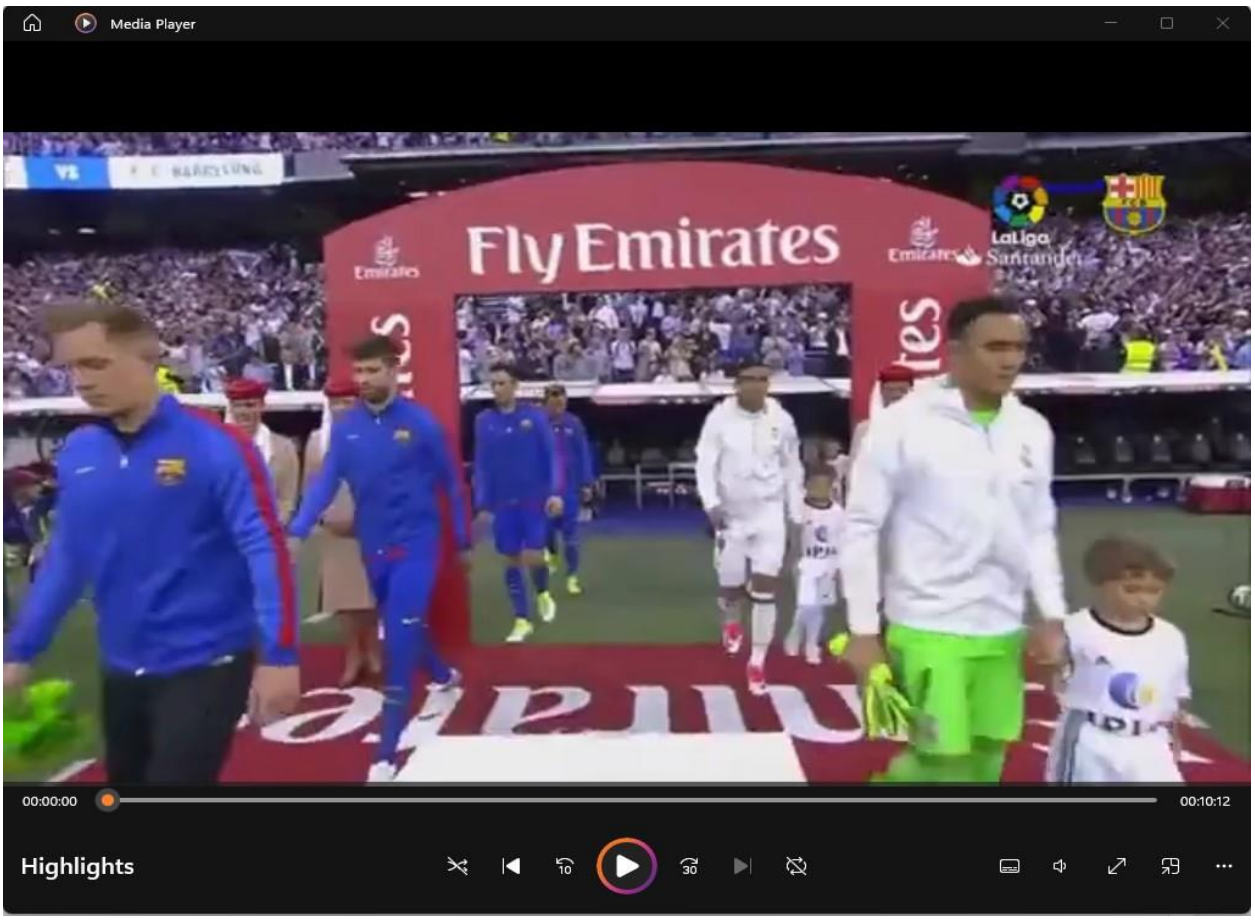
The original video file with length 1 hr 43 minutes:

The output video with 15 minutes length:

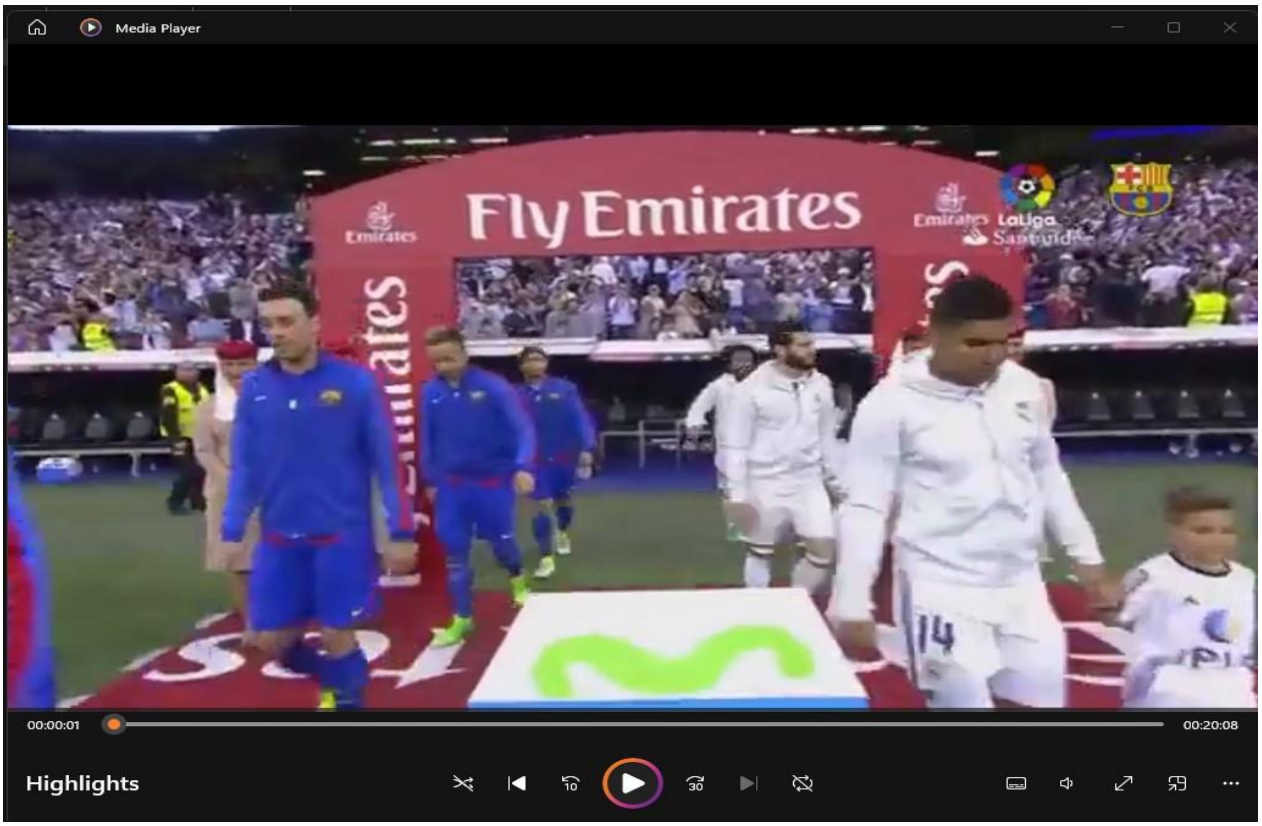


Other sample outputs:

Output with length 10 minutes:



Output with length 20 minutes:



CHAPTER 5

SOURCE CODE

```
#installing ttkthemes for GUI
!pip install ttkthemes
#installing ttkbootstrap for GUI
!pip install ttkbootstrap
#installing librosa for audio processing
!pip install librosa
#installing matplotlib
!pip install matplotlib
#installing IPython
!pip install IPython
##installing numpy
!pip install numpy
#installing pandas
!pip install pandas
#installing moviepy for video processing
!pip install moviepy

#importing libraries for GUI
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
from ttkbootstrap import Style
from ttkthemes import ThemedStyle

#importing libraries for higlights generater
import librosa
import matplotlib.pyplot as plt
import IPython.display as ipd
import numpy as np
import pandas as pd
import os,shutil
from moviepy.video.io.ffmpeg_tools import ffmpeg_extract_subclip
```

```
from moviepy.editor import concatenate_videoclips, VideoFileClip
from math import ceil
import sys

#creating GUI window
window = tk.Tk()

#defining function for upload_button
def upload_video():
    #definging global file_path
    global file_path
    #getting video file selection from user
    file_path = filedialog.askopenfilename(filetypes=[("Video Files", ".mp4;.avi;*.mkv")])
    if file_path:
        # Do something with the uploaded video file
        print("Selected video file:", file_path)
        #changing label to path of selected video
        label.config(text="Selected video: " + file_path)

#defining function for play_button
def play_video():
    #changing label to path of output video
    result = result_label["text"].replace("Summary Result: ", "")
    if os.path.exists(result):
        # Open the generated video in a video player
        if sys.platform.startswith('win32'):
            os.startfile(result)
        elif sys.platform.startswith('linux') or sys.platform.startswith('darwin'):
            subprocess.call(['xdg-open', result])

#defining function for generate button
def select_summary_length():
    #getting summary length as option from input field
    option = summary_length.get()
    print("option", option)
    #converting option to integer
```

```
    option=int(option)
    #deciding correct multiplier
    if option==30:
        mul=1.2
    elif option==20:
        mul=1.8
    elif option==15:
        mul=2.4
    elif option==10:
        mul=3
    else:
        mul=1
    #calling funciton to generate the highlights
    result=summary(mul)
    print(result)
    #changing label to path of output video
    result_label.config(text="Summary Result: " + result)
    # Enable the Play Video button
    play_button.config(state=tk.NORMAL)

def summary(mul):
    # Provide the path to your video file file
    video_path=file_path
    video = VideoFileClip(video_path)

    #extarcting audio of video
    audio = video.audio
    audio_path = 'videoplayback.mp3'

    #writing mp3 file of audio
    audio.write_audiofile(audio_path, codec='mp3')

    # Load the audio file
    audio_data, sample_rate = librosa.load(audio_path)

    # Print the audio data and sample rate
```

```
    print('Audio Data:', audio_data)
    print('Sample Rate:', sample_rate)

        #Breaking down video into chunks of 5 seconds to find energy rise
    chunk_size=5
    window_length = chunk_size * sample_rate

    #seeing an audio sample and it's time-amplitude graph
    a=audio_data[5*window_length:6*window_length]
    ipd.Audio(a, rate=sample_rate)
    energy = sum(a ** 2) / len(a)
    print(energy)
    fig = plt.figure(figsize=(14, 8))
    ax1 = fig.add_subplot(211)
    ax1.set_xlabel('Time')
    ax1.set_ylabel('Amplitude')
    ax1.plot(a)

    #Plotting short time energy distribution histogram of all chunks
    energy = np.array([sum(abs(audio_data[i:i+window_length]**2)/len(audio_data[i:i+window_length])) for
i in range(0, len(audio_data), window_length)])
    plt.hist(energy)
    plt.show()

    # Assuming you already have the array of energy values called 'energy'
    average_energy = np.mean(energy)
    threshold = average_energy
    print(threshold)

    #Finding and setting threshold value of commentator and audience noise above which we want to include
portion in highlights.
    df=pd.DataFrame(columns=['energy','start','end'])

    #setting value of inceremenTer
    inc=0
    if ceil(np.mean(energy)*1000)>2:
```

```
    inc=0.2
thresh=np.mean(energy)*mul+inc
row_index=0

for i in range(len(energy)):
    value=energy[i]
    if(value>=thresh):
        i=np.where(energy == value)[0]
        df.loc[row_index,'energy']=value
        df.loc[row_index,'start']=i[0] * 5
        df.loc[row_index,'end']=(i[0]+1) * 5
        row_index= row_index + 1

#Merge consecutive time intervals of audio clips into one.
temp=[]
i,j,n=0,0,len(df) - 1
while(i<n):
    j=i+1
    while(j<=n):
        if(df['end'][i] == df['start'][j]):
            df.loc[i,'end'] = df.loc[j,'end']
            temp.append(j)
            j=j+1
        else:
            i=j
            break
df.drop(temp,axis=0,inplace=True)

#Extracting subclips from the video file on the basis of energy profile obtained from audio file.
start=np.array(df['start'])
end=np.array(df['end'])

#Create temporary folder for storing subclips generated. This folder will be deleted later after highlights
are generated.
cwd=os.getcwd()
sub_folder=os.path.join(cwd,"Subclips")
```

```
    if os.path.exists(sub_folder):
        shutil.rmtree(sub_folder)
        path=os.mkdir(sub_folder)
    else:
        path=os.mkdir(sub_folder)

    #Extract moments from videos to be added in highlight
    # Assuming you have a DataFrame named df containing the start and end times
    for i in range(len(df)):
        if i != 0:
            # Assuming that noise starts after the shot, so set start point as t-5 seconds to include the shot/wicket action.
            start_lim = start[i] - 5
        else:
            start_lim = start[i]
        end_lim = end[i]
        filename = "highlight" + str(i+1) + ".mp4"
        # Update the video path here
        target_path = sub_folder+'/' + filename
        #finding duration of each subclip generated
        duration = end_lim-start_lim
        #trimming subclips that have length too much
        if duration > 10 and duration <=20:
                ffmpeg_extract_subclip(video_path, start_lim, start_lim+min(duration,15), targetname=target_path)
        elif  duration > 20:
                ffmpeg_extract_subclip(video_path, start_lim+5, start_lim+20, targetname=target_path)
        else:
                ffmpeg_extract_subclip(video_path, start_lim, end_lim, targetname=target_path)

    #getting subclips from the temporary folder
    files = os.listdir(sub_folder)
    files = [sub_folder + "/highlight" + str(i+1) + ".mp4" for i in range(len(files))]
    if len(files) > 0:
        files = [file for file in files if os.path.isfile(file)]
        if len(files) > 0:
```

```
        clips = []
        for file in files:
            clip = VideoFileClip(file)
            #appending video subclips
            clips.append(clip)
        final_clip = concatenate_videoclips(clips)
        #writing final output file
        final_clip.write_videofile("Highlights.mp4", audio_codec='mp3')
        #closing all opened video subclips
        for clip in clips:
            clip.close()
        #deleting the temporary folder created
        shutil.rmtree(sub_folder)
        #returning the output file path to GUI
        #deleting audio file created
        os.remove( 'videoplayback.mp3')
        return "Highlights.mp4"

#Select any one of the themes from below
#style = Style(theme="cyborg")
#style = Style(theme="darkly")
#style = Style(theme="flatly")
#style = Style(theme="pulse")
style = Style(theme="superhero")
#style = ThemedStyle(window)
#style.set_theme("clam")
#style.set_theme("alt")
#style.set_theme("default")
#style.set_theme("classic")

#creating play button to play video
play_button = tk.Button(window, text="Play Video", command=play_video, state=tk.DISABLED)
#aligning play_button
play_button.place(x=165, y=180)

#creating label
```

```
result_label = tk.Label(window, text="Summary Result: ")
result_label.pack()
#aligning label
result_label.place(x=150, y=160)

#setting window title
window.title("Sports Video Summary Generator")
#setting window size
window.geometry("400x240")
#creating label
label = tk.Label(window, text="No video selected")
label.pack()
#aligning label
label.place(x=150,y=50)

#creating label
summary_label = tk.Label(window, text="Select Summary Length")
summary_label.pack()
summary_length = tk.StringVar()

#creating drop down list
summary_combobox = ttk.Combobox(window, textvariable=summary_length, values=["10", "15", "20",
"30"])
summary_combobox.current(0)
summary_combobox.pack()
#creating upload button to upload video
upload_button = tk.Button(window, text="Upload Video", command=upload_video)
#aligning upload button
upload_button.place(x=160, y=70)
#creating generate button to generate highlights video
summary_button = tk.Button(window, text="Generate", command=select_summary_length)
#aligning generate button
summary_button.place(x=170, y=120)

# Start the GUI event loop
window.mainloop()
```

CHAPTER 6

FUTURE ENHANCEMENTS

1. Customizable Highlight Themes:

Allow users to choose from a range of predefined highlight themes, such as "Goals," "Defensive Plays," or "Celebrations." Each theme could dynamically adjust the algorithm's criteria for identifying and highlighting specific moments, providing users with a tailored viewing experience.

2. Multi-Sport Support:

Diverse Sports Analysis: Extend the system's capabilities to accommodate a range of sports, tailoring audio analysis and highlight generation algorithms to specific sports' dynamics.

3. Live Highlight Generation:

Explore real-time highlight generation for live matches. Implement a feature that enables users to generate highlights while a match is in progress, providing an instantaneous way to capture and relive exciting moments as they unfold.

4. Interactive Highlight Editing:

Enhance the application by offering users the ability to manually adjust or fine-tune the extracted highlights. This could include features for trimming, reordering, or adding annotations to specific segments, granting users more control over the final highlights video.

5. Enhanced Audio Analysis:

Speaker Recognition: Implement speaker recognition to differentiate between commentator voices and crowd noise, allowing for more precise highlight selection.

Emotion Detection: Integrate emotion detection algorithms to identify emotional peaks in commentary, capturing the essence of critical moments.

CHAPTER 7
CONCLUSION

The "Football Highlights Generator" embodies a fusion of technology and passion for sports, delivering an innovative solution that transforms the way football enthusiasts engage with match content. Through the integration of audio and video processing techniques, this application empowers users to create personalized and captivating video highlights that encapsulate the excitement of football matches.

The project's journey began with the design and development of an intuitive graphical user interface (GUI) that simplifies the highlight generation process. Users can effortlessly upload their preferred football match videos, select the desired summary length, and obtain a dynamic highlights reel that captures the pivotal moments of the game. The core algorithmic processes behind the "Football Highlights Generator" leverage audio energy analysis to identify key segments within the uploaded video. By calculating energy thresholds and intelligently merging segments, the application produces seamless and engaging highlights that resonate with users' preferences.

The project's significance extends beyond its technical achievements. It reflects the evolving landscape of sports consumption, where modern audiences seek tailored and immersive experiences. The "Football Highlights Generator" caters to this demand, allowing users to curate their own video summaries and relive the thrill of matches without the need for extensive time commitment.

In conclusion, the "Football Highlights Generator" represents a harmonious blend of technology, user-centered design, and the universal appeal of sports. It demonstrates the potential of leveraging audio and video processing to create engaging content that resonates with sports enthusiasts. The project's success lies in its ability to provide a seamless and enjoyable experience, reminding us that the fusion of technology and passion can redefine the way we experience and celebrate our favorite sports moments.

REFERENCES

1. IEEE Colloquium on 'Digital Audio Signal Processing' (Digest No.107).
2. Audio processing with channel filtering using DSP techniques.
3. IEEE Frequency Domain Analysis for Audio Data Forgery Detection

4. Zhou, L., Xu, C., & Lin, W. (2003). Summarization of Soccer Videos for Highlights. In Proceedings of IEEE International Conference on Multimedia and Expo (ICME).

5. https://github.com/

6. https://chat.openai.com/

7. https://bard.google.com/

8. Nwe, T. L., Ho, Y. K., & Zhang, H. J. (2006). Audio based event detection for soccer video. In Proceedings of ACM Multimedia.

9. https://www.w3schools.com/