CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

YouTube can be a distraction. Students may be tempted to watch non-educational videos or become side-tracked by recommended content, which can take away from their study time. There might also be individuals who do not have the time or desire to watch the entire video but still want to learn about its content.

This mini-project solves the above-mentioned problems through the development of a desktop application. The YouTube Transcript Summarizer is a tool that helps users quickly understand the main points of a YouTube video by generating a summary of its transcript.

1.2 OBJECTIVES

Once developed, the application will

- Allow user to input a URL of a YouTube video.
- Create a label and an input field for the user to enter the URL of the YouTube video.
- A function retrieves the URL from the input field, extracts the video ID from the URL, and calls the fetch_transcript() and summarize_transcript() functions
- The above functions are used to retrieve and summarise the transcript. The summary is then displayed to the user.

1.3 METHODOLOGY TO BE FOLLOWED

- Imports the necessary libraries: tkinter for creating a very interactive graphical user interface (GUI), YouTubeTranscriptApi for retrieving the transcript of a YouTube video, and transformers for generating a summary of the transcript.
- Creates the main window for the GUI and configures it with a title and size.
- Creates a label and an input field for the user to enter the URL of the YouTube video.
- Defining the fetch_transcript() function, which retrieves the transcript of a YouTube video using the YouTubeTranscriptApi.get_transcript() function and returns it as a single string.
- Defining the summarize_transcript() function, which uses the transformers library to generate a summary of the input transcript. The summary is generated in chunks of 1000 characters to avoid exceeding the maximum input length for the model.
- Defining the summarize() function, which is called when the user clicks the "Summarize" button. This function retrieves the URL from the input field, extracts the video ID from the URL, and calls the fetch_transcript() and summarize_transcript() functions to retrieve and summarize the transcript. The summary is then displayed in the summary_text widget.
- Creates a button for the user to trigger the summarization process.
- Creates a Scrollbar widget to allow the user to scroll through the summary text if it is too long to fit in the summary_text widget.
- Creates a Text widget and configures it to display the summary text.
- Configures the Scrollbar to control the summary_text widget.
- Runs the main loop for the GUI.

1.4 EXPECTED OUTCOMES

- The code allows users to enter the URL of a YouTube video and generates a summary of its transcript, which is displayed in the GUI.
- The summary text can be scrolled through using the Scrollbar widget if it is too long to fit in the summary_text widget.

1.5 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware

- Operating System: Either Windows 10/ MacOS
- Processor: x86_64-bit_CPU (Intel/AMD) or Mac M1
- 4 Gigabits of Random Access Memory
- 5 Gigabits free disk st

Software

- Python IDLE, PyCharm, Anaconda or VS Code.

CHAPTER 2

FUNDAMENTALS OF PYTHON

2.1 INTRODUCTION TO PYTHON

Python is a high-level, interpreted programming language that is widely used for web development, data analysis, artificial intelligence, and scientific computing. It is known for its simplicity, readability, and flexibility, which make it an ideal language for beginners as well as experienced programmers. Some of the key features of Python include:

- A large standard library that supports many common programming tasks, such as connecting to web servers, reading and writing files, and working with data.
- A simple, easy-to-learn syntax that emphasises readability and reduces the cost of program maintenance.
- An interactive mode that allows users to test code snippets and explore the language in a more interactive way.
- Support for object-oriented, imperative, and functional programming styles.
- Dynamically-typed, which means that users do not need to specify the data type of a variable when declaring it.
- Portable, which means that Python programs can run on any platform that supports the language.

2.2 ADVANTAGES OF PYTHON

- Presence of third-party modules
- Extensive support libraries (NumPy for numerical calculations, Pandas for data analytics, etc.)
- Open source and large active community base
- Versatile, Easy to read, learn and write
- User-friendly data structures
- High-level language
- Dynamically typed language (No need to mention data type based on the value assigned, it takes data type)
- Object-Oriented and Procedural Programming language
- Portable and Interactive
- Ideal for prototypes – provide more functionality with less coding
- Highly Efficient (Python's clean object-oriented design provides enhanced process control, and the language is equipped with excellent text processing and

integration capabilities, as well as its own unit testing framework, which makes it more efficient.)

- Internet of Things (IoT) Opportunities
- Interpreted Language
- Portable across Operating systems

2.3 PYTHON LISTS

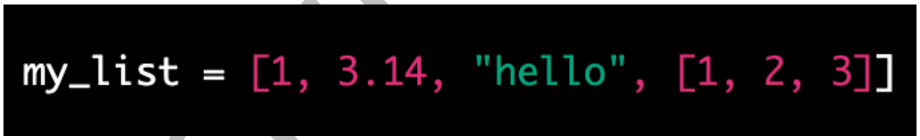
Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionaries all with different qualities and usage.

Lists are created using square brackets.

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.



```
my_list = [1, 3.14, "hello", [1, 2, 3]]
```

Fig 2.3

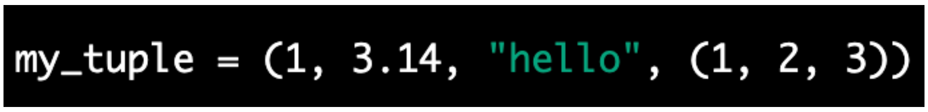
2.4 PYTHON TUPLES

A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.

Creating a Tuple

A tuple is created by placing all the items (elements) inside parentheses (), separated by commas. The parentheses are optional however, it is a good practice to use them.

A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).



```
my_tuple = (1, 3.14, "hello", (1, 2, 3))
```

Fig 2.4

2.5 PYTHON SETS

A set is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed).

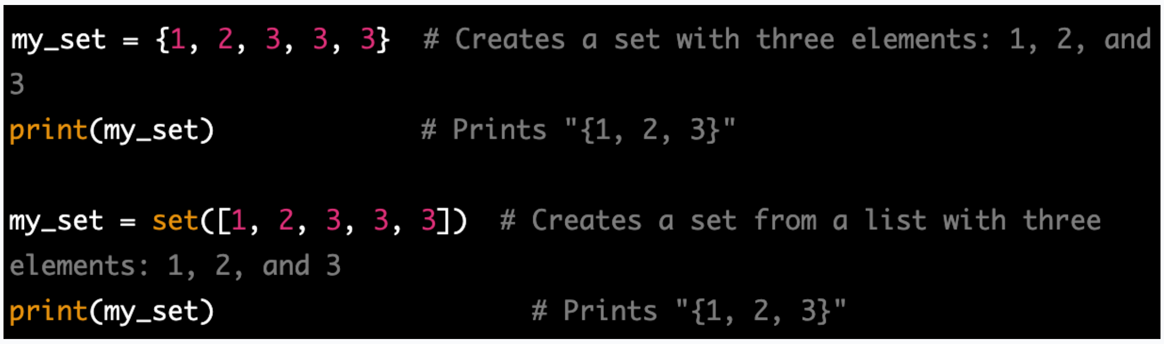
However, a set itself is mutable. We can add or remove items from it.

Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc.

Creating Python Sets

A set is created by placing all the items (elements) inside curly braces {}, separated by comma, or by using the built-in set() function.

It can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have mutable elements like lists, sets or dictionaries as its elements.



```
my_set = {1, 2, 3, 3, 3}  # Creates a set with three elements: 1, 2, and
3
print(my_set)             # Prints "{1, 2, 3}"

my_set = set([1, 2, 3, 3, 3])  # Creates a set from a list with three
elements: 1, 2, and 3
print(my_set)                  # Prints "{1, 2, 3}"
```

Fig 2.5

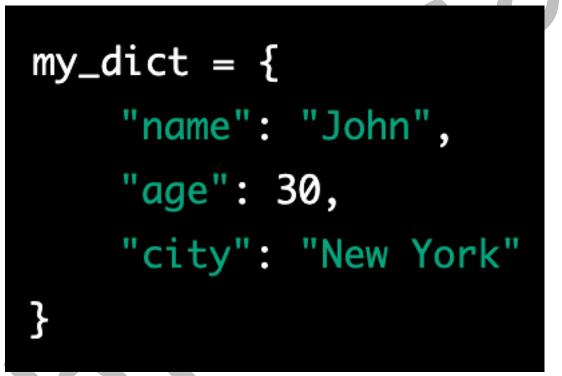
2.6 PYTHON DICTIONARIES

Dictionaries are used to store data values in key: value pairs. A dictionary is a collection which is ordered*, changeable and do not allow duplicates. Dictionaries are written with curly brackets, and have keys and values.

Dictionary Items:

Dictionary items are ordered, changeable, and does not allow duplicates.

Dictionary items are presented in key: value pairs, and can be referred to by using the key name.



```
my_dict = {
    "name": "John",
    "age": 30,
    "city": "New York"
}
```

Fig 2.6

2.7 PYTHON FUNCTIONS

A function is a block of code that performs a specific task.

Suppose, you need to create a program to create a circle and colour it. You can create two functions to solve this problem:

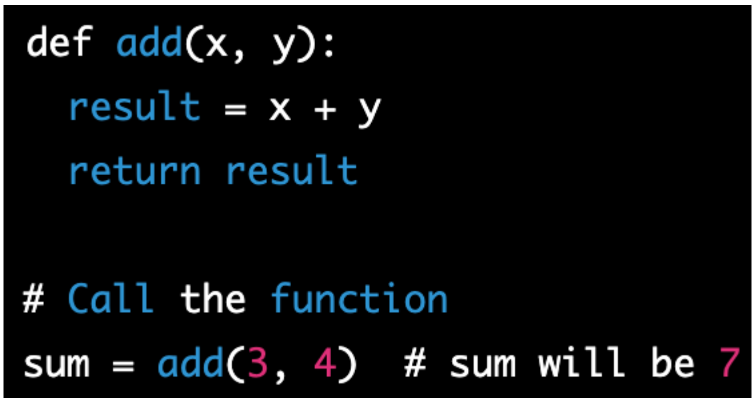
- create a circle function
- create a color function

Dividing a complex problem into smaller chunks makes our program easy to understand and reuse.

Types of function

There are two types of function in Python programming:

- **Standard library functions** - These are built-in functions in Python that are available to use.
- **User-defined functions** - We can create our own functions based on our requirements.



```
def add(x, y):
  result = x + y
  return result

# Call the function
sum = add(3, 4)  # sum will be 7
```

Fig 2.7

CHAPTER 3

FUNDAMENTALS OF TKINTER

3.1 INTRODUCTION

Tkinter is a Python package that provides a interface to the Tk GUI toolkit. It is built into Python and provides a relatively easy-to-use interface for creating and laying out a graphical user interface (GUI).

Here are some fundamental concepts that you should understand when using Tkinter:

- **Widgets**: Widgets are the basic building blocks of a GUI. Tkinter provides several types of widgets, including buttons, labels, and text entry fields.

- **Containers**: Containers are widgets that can contain other widgets. For example, a frame is a container that can hold other widgets.
- **Layouts**: Layouts are used to arrange widgets within a container. Tkinter provides several layout managers, such as the pack and grid managers, that can be used to organize widgets in a container.
- **Events**: Events are actions that occur within a GUI, such as a user clicking a button or entering text into a text field. Tkinter allows you to bind code to specific events, so that you can specify what should happen when an event occurs.

These are **19 widgets** available in Python Tkinter module.

3.2 WIDGETS

There are various controls, such as **buttons**, **labels**, **scrollbars**, **radio buttons**, and **text boxes** used in a GUI application. These **little components** or controls of **Graphical User Interface (GUI)** are known as **widgets** in Tkinter.

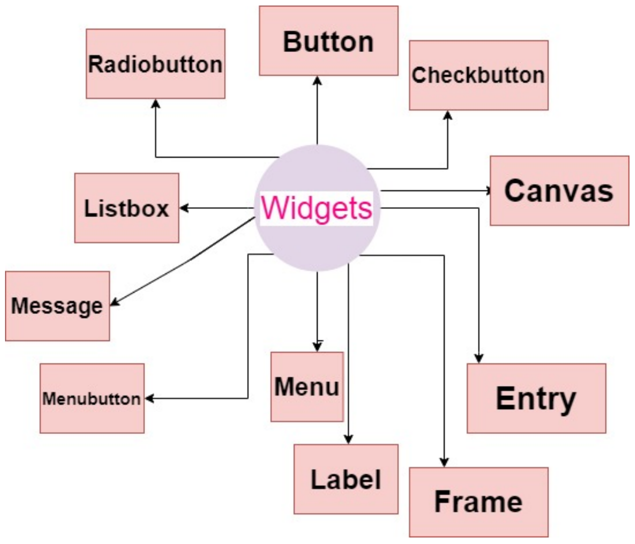


Fig 3.2

3.3 LABLES

A label in tkinter is a widget that displays text or an image. It is often used to display static text or to label other widgets. Labels can be created using the tkinter Label() constructor and can be configured with various options such as font, color, and size. Labels can also be aligned within their parent widget using the anchor option. Labels can be updated by changing their text or image attribute.

3.4 FRAME

The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way. It works like a container, which is responsible for arranging the position of other widgets.

It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets. A frame can also be used as a foundation class to implement complex widgets.

3.5 BUTTONS

The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

Syntax

Here is the simple syntax to create this widget:

w= Button (master, option=value, …)

CHAPTER 4

DESIGN

4.1 DESIGN GOALS

- This mini project has ensured that the user has an interactive and explorable environment.
- The interface is user friendly, simple to understand and has tried to ensure that there are no bugs.
- This project is fast and secure.
- Easy to access.

4.2 ALGORITHM

This code creates a graphical user interface (GUI) using the Tkinter library. When the user enters a YouTube URL in the input field and clicks the "Summarize" button, the following steps are performed:

- The video's ID is extracted from the URL by splitting the string on the '=' character and taking the second element of the resulting list.
- The fetch_transcript function is called with the video ID as an argument. This function retrieves the transcript of the video from the YouTubeTranscriptApi and returns it as a string.
- The summarize_transcript function is called with the transcript string as an argument. This function uses the transformers library to create a summarization pipeline using the BERT (Bidirectional Encoder Representations from Transformers) model and processes the transcript through the pipeline. The summary is returned as a string.
- The text widget is cleared and the summary string is inserted into it.

The GUI also includes a Scrollbar widget that can be used to scroll through the summary text if it exceeds the size of the text widget.

4.3 FLOWCHART

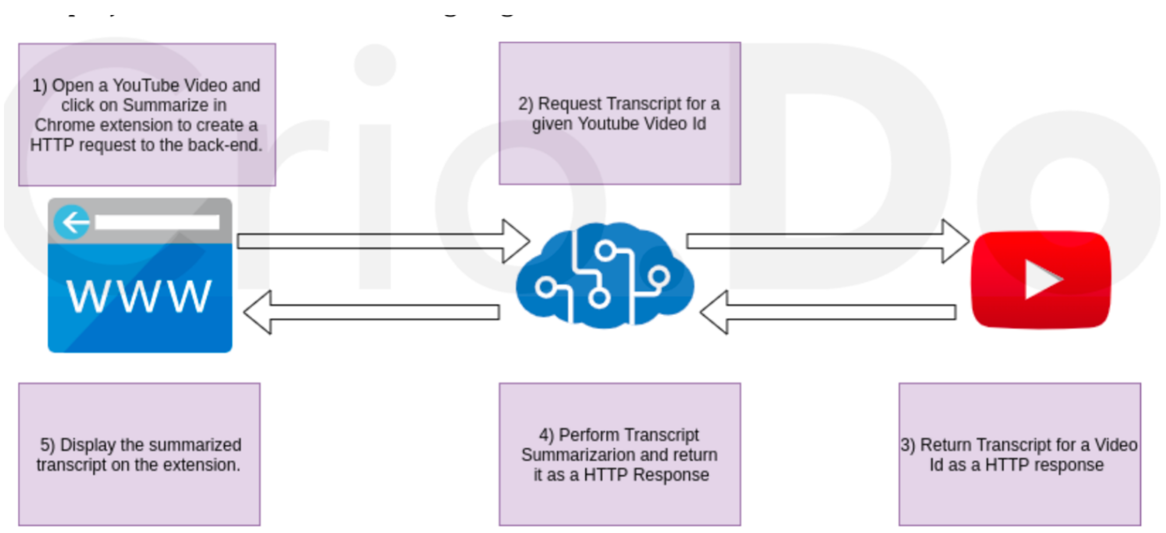
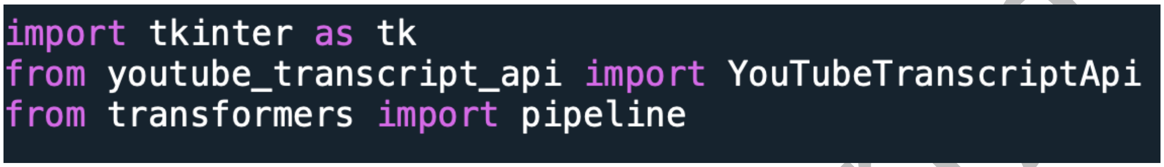

1) Open a YouTube Video and click on Summarize in Chrome extension to create a HTTP request to the back-end.
2) Request Transcript for a given Youtube Video Id
WWW
3) Return Transcript for a Video Id as a HTTP response
4) Perform Transcript Summarizarion and return it as a HTTP Response
5) Display the summarized transcript on the extension.

Fig 4.3

CHAPTER 5

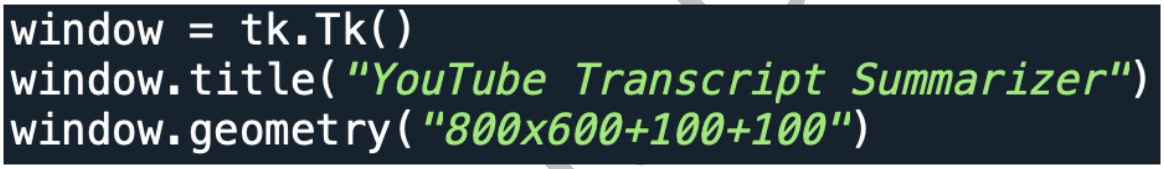
IMPLEMENTATION

MODULE 5.1 LIBRARIES



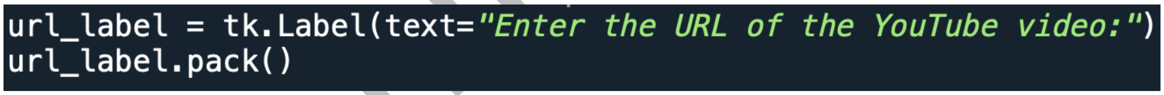
```
import tkinter as tk
from youtube_transcript_api import YouTubeTranscriptApi
from transformers import pipeline
```

MODULE 5.2 Create the main window



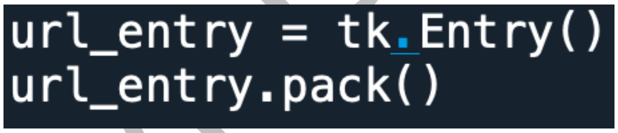
```
window = tk.Tk()
window.title("YouTube Transcript Summarizer")
window.geometry("800x600+100+100")
```

MODULE 5.3 Create a label for the URL input



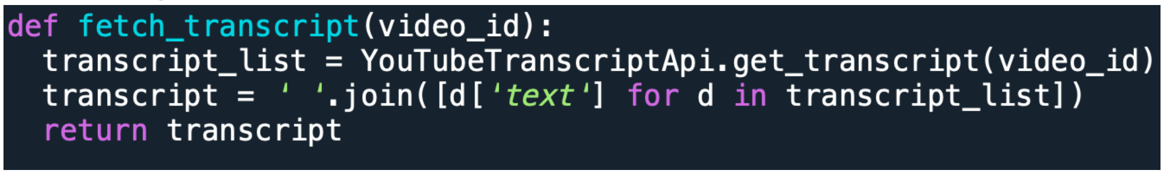
```
url_label = tk.Label(text="Enter the URL of the YouTube video:")
url_label.pack()
```

MODULE 5.4 Create an input field for the URL



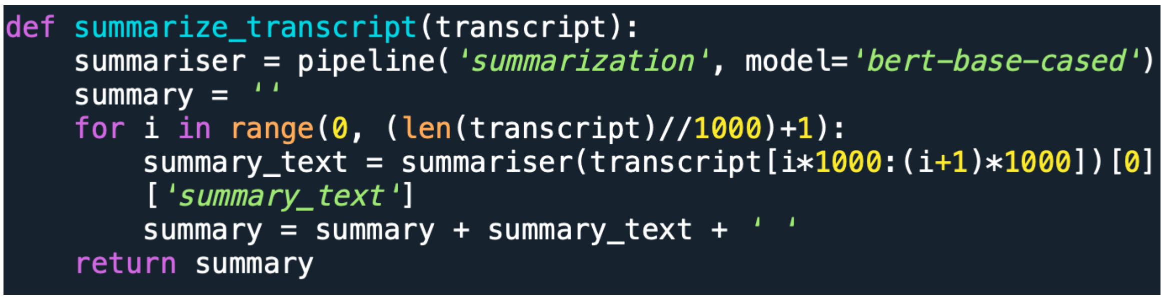
```
url_entry = tk.Entry()
url_entry.pack()
```

MODULE 5.5 Fetching Transcript



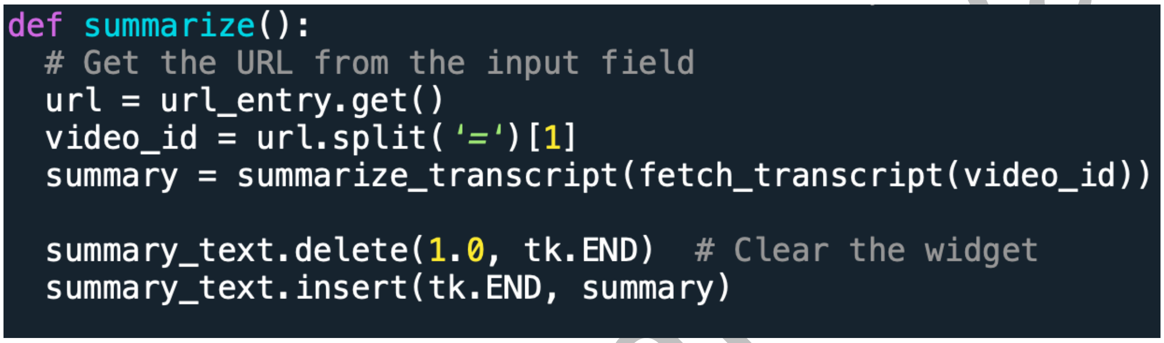
```
def fetch_transcript(video_id):
  transcript_list = YouTubeTranscriptApi.get_transcript(video_id)
  transcript = ' '.join([d['text'] for d in transcript_list])
  return transcript
```

MODULE 5.6 Summarizing the Transcript



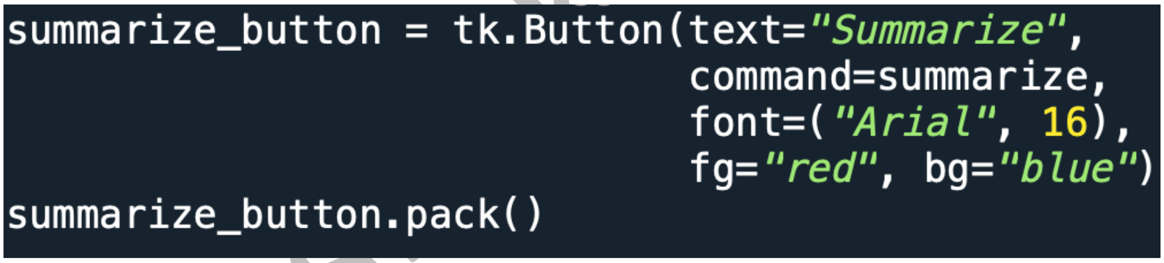
```
def summarize_transcript(transcript):
    summariser = pipeline('summarization', model='bert-base-cased')
    summary = ''
    for i in range(0, (len(transcript)//1000)+1):
        summary_text = summariser(transcript[i*1000:(i+1)*1000])[0]
        ['summary_text']
        summary = summary + summary_text + ' '
    return summary
```

MODULE 5.7 Create a function to summarize the transcript



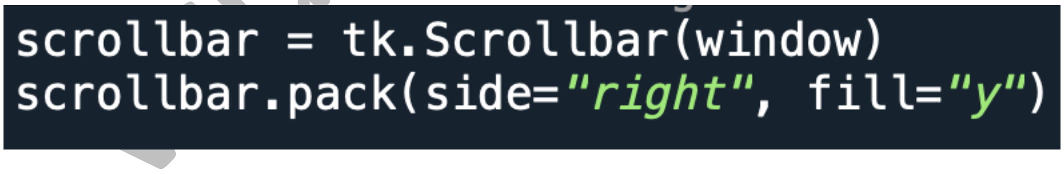
```
def summarize():
  # Get the URL from the input field
  url = url_entry.get()
  video_id = url.split('=')[1]
  summary = summarize_transcript(fetch_transcript(video_id))

  summary_text.delete(1.0, tk.END)  # Clear the widget
  summary_text.insert(tk.END, summary)
```

MODULE 5.8 Create a button to trigger the summarization



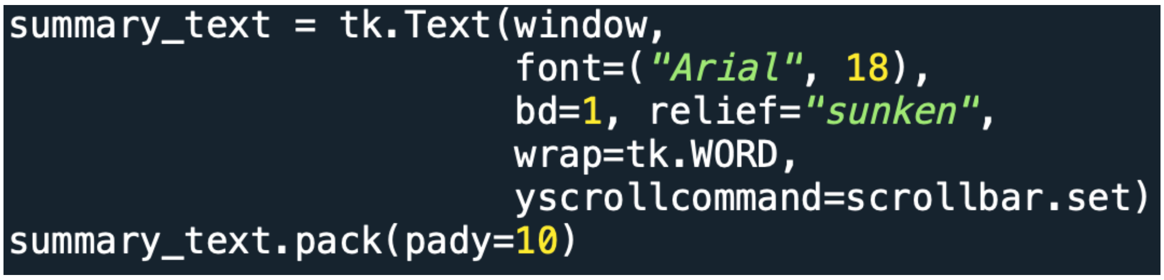
```
summarize_button = tk.Button(text="Summarize",
                             command=summarize,
                             font=("Arial", 16),
                             fg="red", bg="blue")
summarize_button.pack()
```

MODULE 5.9 Create a Scrollbar widget



```
scrollbar = tk.Scrollbar(window)
scrollbar.pack(side="right", fill="y")
```

MODULE 5.10 Create a Text widget

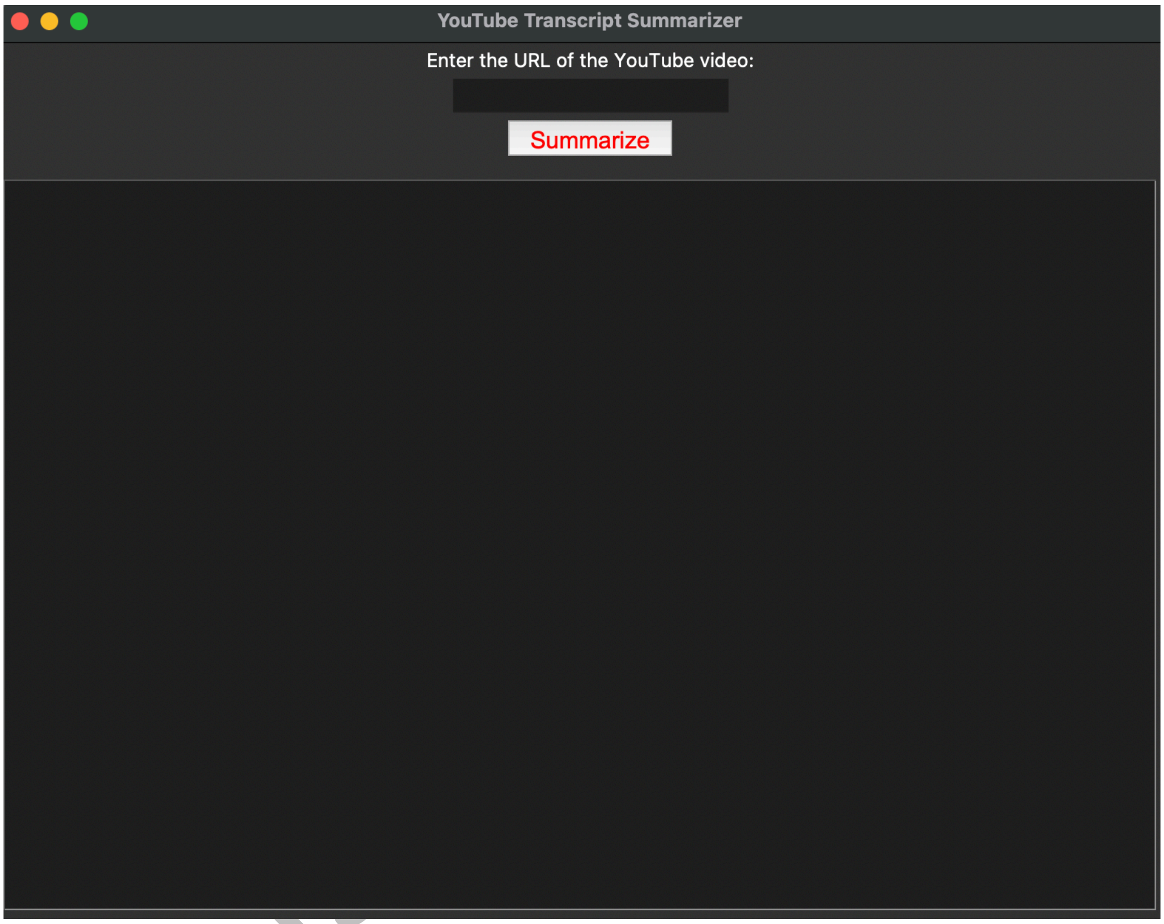


```
summary_text = tk.Text(window,
                       font=("Arial", 18),
                       bd=1, relief="sunken",
                       wrap=tk.WORD,
                       yscrollcommand=scrollbar.set)
summary_text.pack(pady=10)
```

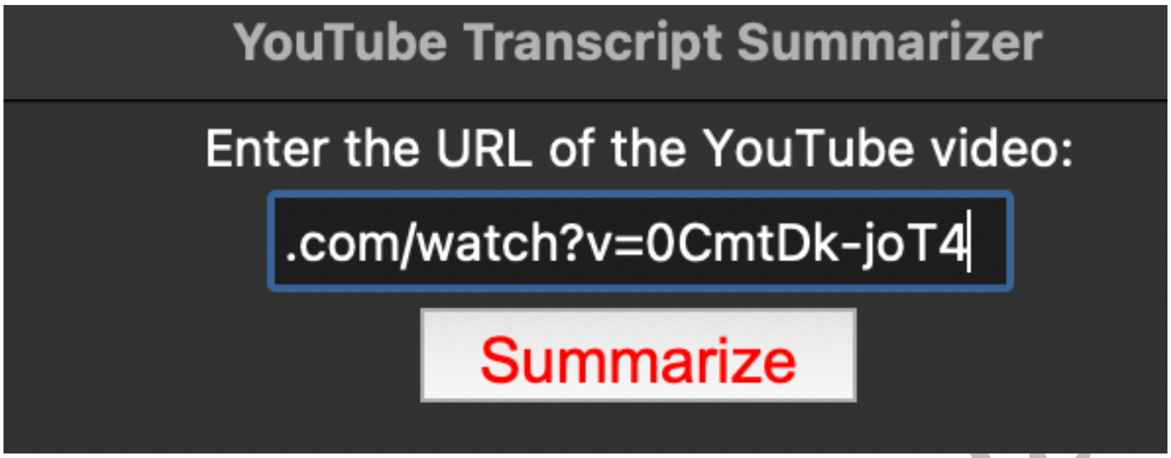
CHAPTER 6

RESULTS

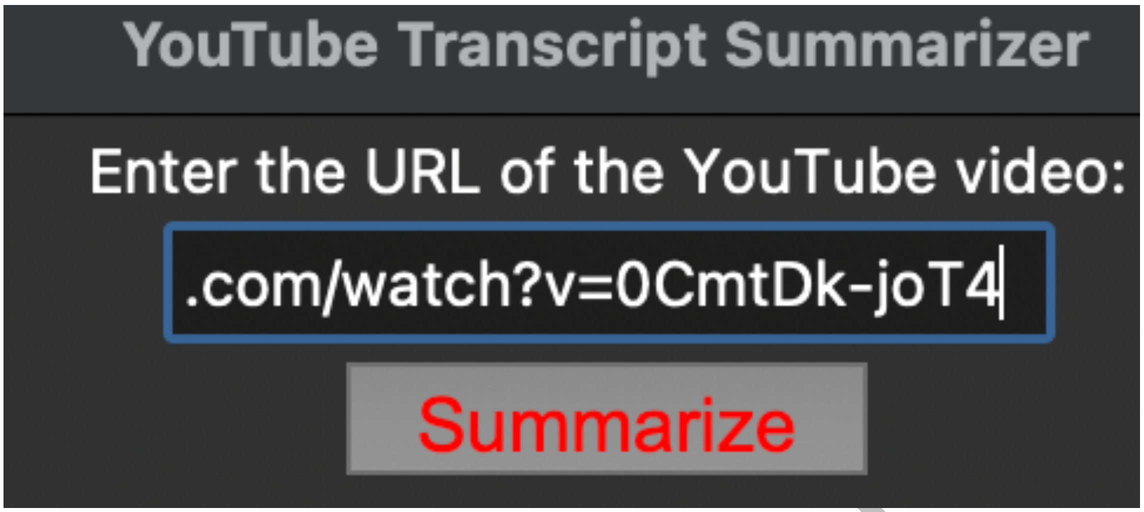
6.1 MAIN SCREEN


YouTube Transcript Summarizer
Enter the URL of the YouTube video:
Summarize

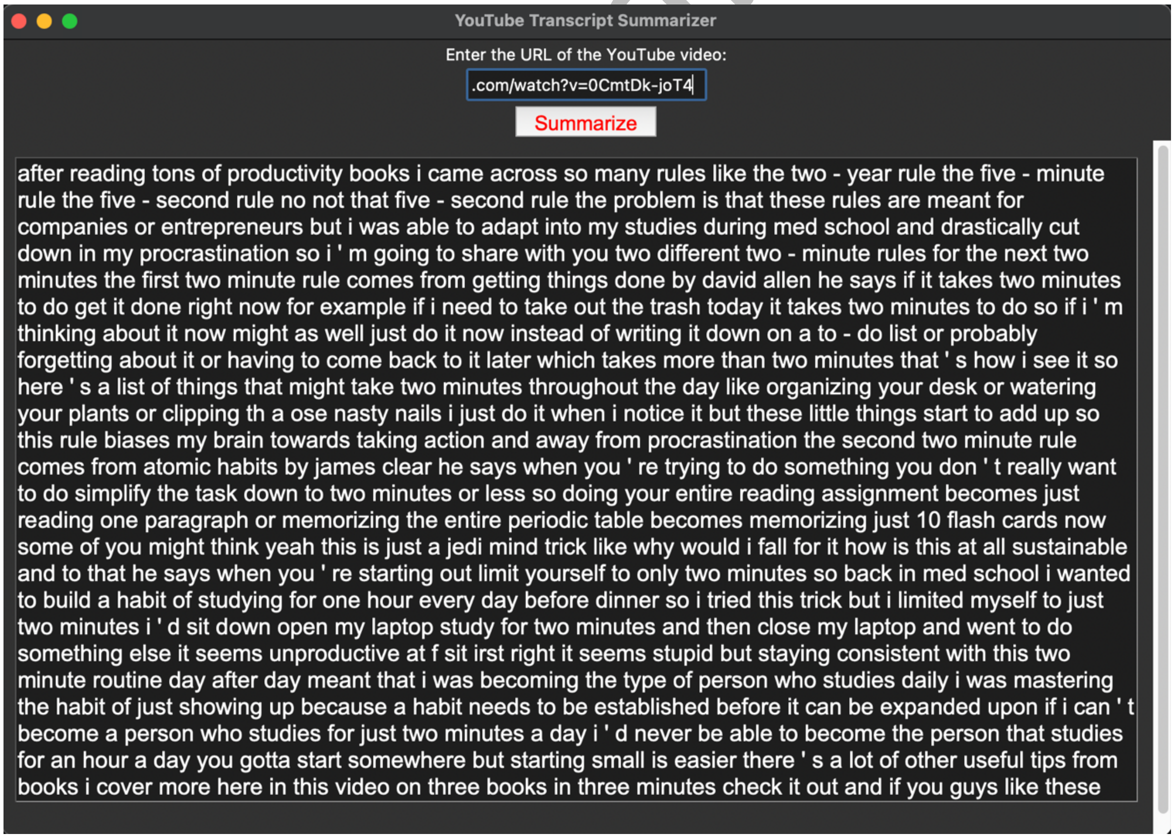
6.2 URL ENTRY


YouTube Transcript Summarizer
Enter the URL of the YouTube video:
.com/watch?v=0CmtDk-joT4
Summarize

6.3 Click Summarize


YouTube Transcript Summarizer
Enter the URL of the YouTube video:
.com/watch?v=0CmtDk-joT4
Summarize

6.4 Summary


YouTube Transcript Summarizer
Enter the URL of the YouTube video:
.com/watch?v=0CmtDk-joT4
Summarize

```
after reading tons of productivity books i came across so many rules like the two - year rule the five - minute
rule the five - second rule no not that five - second rule the problem is that these rules are meant for
companies or entrepreneurs but i was able to adapt into my studies during med school and drastically cut
down in my procrastination so i ' m going to share with you two different two - minute rules for the next two
minutes the first two minute rule comes from getting things done by david allen he says if it takes two minutes
to do get it done right now for example if i need to take out the trash today it takes two minutes to do so if i ' m
thinking about it now might as well just do it now instead of writing it down on a to - do list or probably
forgetting about it or having to come back to it later which takes more than two minutes that ' s how i see it so
here ' s a list of things that might take two minutes throughout the day like organizing your desk or watering
your plants or clipping th a ose nasty nails i just do it when i notice it but these little things start to add up so
this rule biases my brain towards taking action and away from procrastination the second two minute rule
comes from atomic habits by james clear he says when you ' re trying to do something you don ' t really want
to do simplify the task down to two minutes or less so doing your entire reading assignment becomes just
reading one paragraph or memorizing the entire periodic table becomes memorizing just 10 flash cards now
some of you might think yeah this is just a jedi mind trick like why would i fall for it how is this at all sustainable
and to that he says when you ' re starting out limit yourself to only two minutes so back in med school i wanted
to build a habit of studying for one hour every day before dinner so i tried this trick but i limited myself to just
two minutes i ' d sit down open my laptop study for two minutes and then close my laptop and went to do
something else it seems unproductive at f sit irst right it seems stupid but staying consistent with this two
minute routine day after day meant that i was becoming the type of person who studies daily i was mastering
the habit of just showing up because a habit needs to be established before it can be expanded upon if i can ' t
become a person who studies for just two minutes a day i ' d never be able to become the person that studies
for an hour a day you gotta start somewhere but starting small is easier there ' s a lot of other useful tips from
books i cover more here in this video on three books in three minutes check it out and if you guys like these
```

CHAPTER 7

CONCLUSION

The mini project has successfully accomplished the goals it had set out in the objectives and design sections of this report.

This UI implements all the modules successfully that are mentioned in this report.

Hence, using this project users quickly understand the main points of a YouTube video by generating a summary of its transcript.