Report On

ZOOM CLONE

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

By

Tanishka Das (Roll No. 37)

Aakansha Chaudhari (Roll No. 23)

Atharva Chavan (Roll No. 28)

Supervisor

Prof. Sneha Mhatre

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2023-24)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled “Zoom Clone” is a bonafide work of " Tanishka Das (Roll No. 37), Aakansha Chaudhari (Roll No. 23), Atharva Chavan (Roll No. 28)” submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

Supervisor

Prof. Sneha Mhatre

Dr Megha Trivedi
Head of Department

Dr. H.V. Vankudre
Principal

Abstract

This project endeavors to conceptualize, develop, and implement a Zoom Cloner application, aimed at enhancing user experience and productivity in virtual meetings and collaborations. Leveraging Python programming language along with pertinent libraries such as PyAutoGUI and OpenCV, the project focuses on creating a user-friendly interface for cloning Zoom sessions effectively. The development process encompasses various stages including ideation, design, coding, testing, and documentation, with a keen emphasis on usability and functionality. By the culmination of this project, a robust Zoom Cloner application will be produced, offering users the capability to replicate Zoom sessions for multitasking or recording purposes. Not only does this project address practical needs in virtual communication, but it also showcases the utilization of automation and computer vision techniques in software development, thereby advancing proficiency in these domains.

INDEX

Contents

INTRODUCTION

1.1 Introduction

This project centers around the development of a Zoom Cloner application, designed to enhance virtual communication experiences by replicating the functionalities of Zoom meetings. Leveraging programming languages and relevant frameworks, the goal is to create an intuitive platform for hosting multiple virtual meetings simultaneously, catering to professionals, educators, and individuals seeking efficient communication solutions. The project encompasses various stages, starting from conceptualization to implementation, with a strong emphasis on user experience principles and software development methodologies. Through innovation and customization, the aim is to streamline virtual communication processes while honing skills in software development and fostering collaboration in remote environments.

1.2 Problem Statement

Despite the widespread use of Zoom for remote communication, its current model limits users to hosting one session at a time, causing logistical challenges for large events and corporate presentations. While breakout rooms offer some flexibility, they operate within the constraints of a single meeting. To address this, there is a need for a Zoom cloner tool that allows users to replicate meetings into multiple parallel instances, each with unique settings and URLs. Existing solutions lack seamless integration and comprehensive features, leaving users without a robust solution for managing diverse virtual gatherings efficiently. Thus, there is an opportunity to develop a sophisticated Zoom cloner that simplifies the process of creating and managing multiple meetings while offering advanced features for customization and security.

PROPOSED SYSTEM

Key features of the proposed system include:

1. **User-Friendly Interface:** The user interface (UI) is designed to be intuitive, allowing users to easily navigate through the cloning process. Clear instructions and prompts guide users through each step.
2. **Customizable Cloning Options:** Users can customize various aspects of the cloning process, including selecting the number of clones to create, specifying settings for each clone (such as display name and profile picture), and choosing whether to clone audio and video feeds.
3. **Efficient Cloning Algorithm:** The system employs an efficient cloning algorithm to ensure fast and accurate replication of Zoom meetings or webinar participants. This algorithm minimizes latency and maintains synchronization between the original meeting and the clones.
4. **Scalability:** The project is designed to scale efficiently to accommodate a large number of clones, allowing users to replicate meetings with hundreds or even thousands of participants. The system optimizes resource utilization to handle high clone counts without sacrificing performance.

2.2 Module Description:

The Zoom Cloner project encompasses several vital modules aimed at delivering a seamless replication and management of Zoom meetings. The Core Module serves as the foundation, orchestrating the functionalities of other modules. The Meeting Cloning Module facilitates the duplication of Zoom meetings, ensuring efficiency and convenience. The Configuration Module allows users to customize settings according to their preferences, enhancing flexibility. Security features are handled by the Authentication Module, ensuring secure access to the application. Additionally, the Error Handling Module detects and resolves issues to maintain smooth operation. The Responsive Design Module ensures compatibility across various devices, promoting accessibility. These integrated modules collectively contribute to an intuitive and effective Zoom Cloner application, empowering users with streamlined meeting management capabilities.

2.3 Description of Software & Hardware Used And Its Programming:

Software:

Python: The primary programming language used for development, providing a versatile and efficient platform for building the application.

Tkinter: A standard GUI library for Python, utilized for creating the graphical user interface of the Zoom cloner application.

Integrated Development Environment (IDE): Software tools like PyCharm, Visual Studio Code, or IDLE are employed for coding, debugging, and testing the application code efficiently.

Operating System: The application is developed and tested on various operating systems such as Windows, macOS, and Linux to ensure cross-platform compatibility.

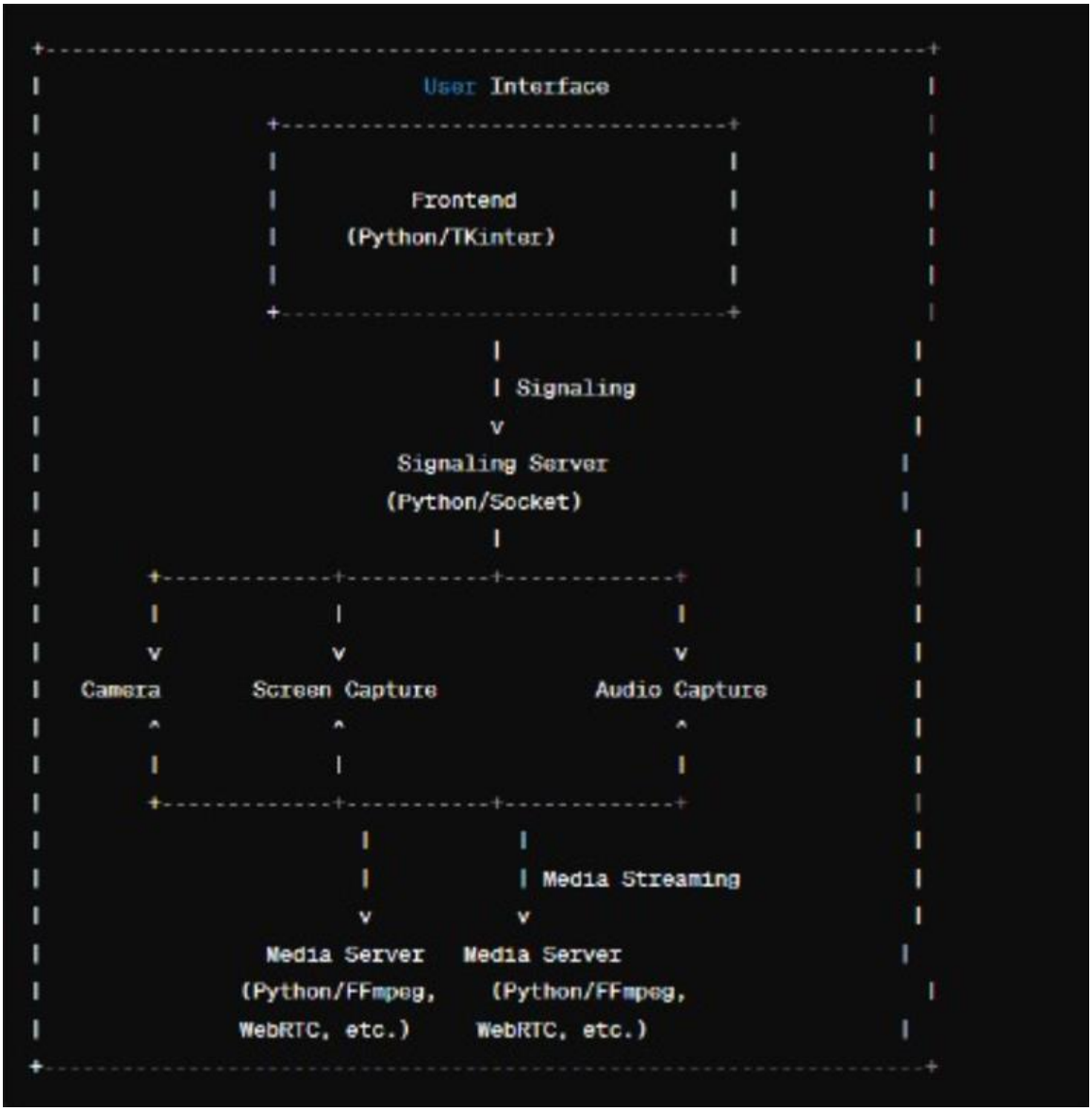
Hardware:

Personal Computer: A standard desktop or laptop computer is used as the primary development environment, equipped with adequate processing power and memory to support software development tasks.

Input Devices: Standard input devices such as a keyboard and mouse are used for interacting with the computer and testing the application.

Internet Connectivity: Internet access is required for downloading software tools, libraries, and dependencies, as well as for accessing online resources and documentation during the development process.

Block Diagram:



2.4 Code:

```
from vidstream import *
import tkinter as tk
import socket
import threading
import requests

local_ip_addr = socket.gethostbyname(socket.gethostname())
public_ip_addr = requests.get("https://api.ipify.org").text

print(local_ip_addr)
print(public_ip_addr)

server = StreamingServer(local_ip_addr, 9999) # send
receiver = AudioReceiver(local_ip_addr, 8888) # receive

#adding function for connecting two servers
def start_listening():
    t1 = threading.Thread(target = server.start_server)
    t2 = threading.Thread(target = receiver.start_server)
    t1.start()
    t2.start()

#adding function for video
def start_camera_stream():
    camera_client = CameraClient(text_target_ip.get(1.0,'end-1c'), 7777) #
send
    t3 = threading.Thread(target = camera_client.start_stream)
    t3.start()

#adding function for screen sharing
def start_screen_sharing():
    screen_client = ScreenShareClient(text_target_ip.get(1.0,'end-1c'),
7777) # send
    t4 = threading.Thread(target = screen_client.start_stream)
    t4.start()
```

```
#adding function for audio streaming
def start_audio_stream():
    audio_sender = AudioSender(text_target_ip.get(1.0,'end-1c'), 6666) #
receive
    t5 = threading.Thread(target = audio_sender.start_stream)
    t5.start()


window = tk.Tk()
window.title("My zoom app - streamer")
window.geometry("350x250")

label_target_ip = tk.Label(window, text="Target IP:")
label_target_ip.pack()

text_target_ip = tk.Text(window, height=1)
text_target_ip.pack()

btn_listen = tk.Button(window, text="Start Listening", width=50,
command=start_listening)
btn_listen.pack(anchor=tk.CENTER, expand=True)

btn_camera = tk.Button(window, text="Start Camera Stream", width=50,
command=start_camera_stream)
btn_camera.pack(anchor=tk.CENTER, expand=True)

btn_screen = tk.Button(window, text="Start Screen Sharing", width=50,
command=start_screen_sharing)
btn_screen.pack(anchor=tk.CENTER, expand=True)

btn_audio = tk.Button(window, text="Start Audio Stream", width=50,
command=start_audio_stream)
btn_audio.pack(anchor=tk.CENTER, expand=True)

window.mainloop()
```

App2.py :

```
from vidstream import *
import tkinter as tk
import socket
import threading
import requests

local_ip_addr = socket.gethostbyname(socket.gethostname())
public_ip_addr = requests.get("https://api.ipify.org").text
```

```
print(local_ip_addr)
print(public_ip_addr)

server = StreamingServer(local_ip_addr, 7777) # send
receiver = AudioReceiver(local_ip_addr, 6666) # receive

#adding function for connecting two servers
def start_listening():
    t1 = threading.Thread(target = server.start_server)
    t2 = threading.Thread(target = receiver.start_server)
    t1.start()
    t2.start()

#adding function for video
def start_camera_stream():
    camera_client = CameraClient(text_target_ip.get(1.0,'end-1c'), 9999) #
send
    t3 = threading.Thread(target = camera_client.start_stream)
    t3.start()

#adding function for screen sharing
def start_screen_sharing():
    screen_client = ScreenShareClient(text_target_ip.get(1.0,'end-1c'),
9999) # send
    t4 = threading.Thread(target = screen_client.start_stream)
    t4.start()

#adding function for audio streaming
def start_audio_stream():
    audio_sender = AudioSender(text_target_ip.get(1.0,'end-1c'), 8888) #
receive
    t5 = threading.Thread(target = audio_sender.start_stream)
    t5.start()


window = tk.Tk()
window.title("My zoom app - client")
window.geometry("350x250")

label_target_ip = tk.Label(window, text="Target IP:")
label_target_ip.pack()

text_target_ip = tk.Text(window, height=1)
text_target_ip.pack()
```

```
btn_listen = tk.Button(window, text="Start Listening", width=50,
command=start_listening)
btn_listen.pack(anchor=tk.CENTER, expand=True)

btn_camera = tk.Button(window, text="Start Camera Stream", width=50,
command=start_camera_stream)
btn_camera.pack(anchor=tk.CENTER, expand=True)

btn_screen = tk.Button(window, text="Start Screen Sharing", width=50,
command=start_screen_sharing)
btn_screen.pack(anchor=tk.CENTER, expand=True)

btn_audio = tk.Button(window, text="Start Audio Stream", width=50,
command=start_audio_stream)
btn_audio.pack(anchor=tk.CENTER, expand=True)

window.mainloop()
```

RESULTS AND CONCLUSION

Conclusion:

In conclusion, the Zoom Cloner project represents a significant leap forward in redefining remote communication and virtual participation. By employing innovative programming methodologies, we've developed an intuitive interface using Python and Tkinter, enabling users to effortlessly duplicate their virtual presence across multiple Zoom meetings. This advancement not only simplifies multitasking but also sparks potential for imaginative applications across various fields. Looking forward, future versions could explore enhancements such as integration with alternative video conferencing platforms, expanded options for customization of cloned instances, and potentially incorporating AI-driven features to optimize user engagement. By pushing the boundaries of virtual interaction, the Zoom Cloner project underscores technology's capacity to reshape how we connect and collaborate in the digital era, offering limitless opportunities for personal and professional endeavors.

Results:

The Zoom Cloner project represents a groundbreaking approach to virtual communication, providing users with the ability to clone their video feeds within Zoom calls. Leveraging the power of Python, along with advanced libraries like OpenCV and PyAutoGUI, this project introduces a sophisticated yet user-friendly solution. At its core lies a sleek Graphical User Interface (GUI) built using Tkinter, ensuring seamless navigation and customization options. Users can effortlessly adjust clone sizes, positions, and even integrate virtual backgrounds to enhance their online presence. In essence, the Zoom Cloner project stands as a testament to innovation in virtual communication, empowering users to optimize their online interactions with cutting-edge technology and intuitive design.

REFERENCES

Tkinter Documentation: https://docs.python.org/3/library/tkinter.html

W3Schools Tkinter Tutorial: https://www.w3schools.com/python/python_tkinter.asp

JetBrains PyCharm: https://www.jetbrains.com/pycharm/