A

Project Report

on

Screen Recoder App

Master of Science in (Computer Science)

दक्षिण बिहार केंद्रीय विश्वविद्यालय

Central University of South Bihar

Under The Supervision of
Dr. Nemi Chandra Rathore
Assistant Professor
Department of Computer Science

Submitted By:

Md. Asif Faizi (CUSB2202312015)

Vikash Kumar (CUSB2202312028)

Department of Computer Science

School Of Mathematics, Statistics & Computer Science
Central University of South Bihar, Gaya

December, 2023

दक्षिण बिहार केंद्रीय विश्वविद्यालय
Central University of South Bihar

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"Screen Recoder App"** in partial fulfillment of the requirements for the completion of the Mini-Project of the **Master of Science in Computer Science** submitted in the Department of Computer Science of Central University of South Bihar, Gaya, is an original work carried out during the period of November, 2023, under the supervision of **Dr. Nemi Chandra Rathore (Assistant Professor),** Department of Computer Science, Central University of South Bihar, Gaya.

The matter presented in the project has not been submitted by us for the award of any other project of this or any other places.

Md. Asif Faizi (CUSB2202312015)

Vikash kumar (CUSB2202312028)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Nemi Chandra Rathore

(Assistant Professor)

ACKNOWLEDGEMENT

We take this occasion to thank God, almighty for blessing us with his grace and takingour endeavor to a successful culmination. We extend our sincere and heartfelt thanks to our esteemed guide,

Dr. **Nemi Chandra Rathore** for providing us with the right guidance and advice at the crucial junctures and for showing us the right way. We extend our sincere thanks to our respected guide **Mr. Purnendu Prabhat,** for Guidance and support to do the project work.

We would like to thank the other faculty members also, at this occasion. Last but not the least; we would like to thank to our team members & friends for the support and encouragement they have given us during the Project Report of our work.

Thank You,
Sincerely,

Md. Asif Faizi (CUSB2202312015)

Vikash kumar (CUSB2202312028)

Table of Contents

Title **Page No.**

Chapter 1
Introduction

1. Introduction

The Simple Screen Recorder is a basic Python application designed to capture the screen and save it as a video file. The project utilizes the PyAutoGUI, OpenCV, and Tkinter libraries to provide a simple graphical user interface for initiating and terminating the screen recording process.

1.1 Formulation of Problem

The primary problem is the absence of a lightweight and intuitive screen recording application that allows users to effortlessly capture and save their on-screen activities. Existing tools may be too feature-rich, requiring users to navigate through complex settings, or too simplistic, lacking essential functionalities. The challenge is to develop a solution that strikes a balance between simplicity and functionality, ensuring that users can easily record their screens without the need for extensive technical knowledge.

Tools and Technology Used

The following technologies and tools were used in the development of the To-Do List App:

I. **Python:**
Python serves as the primary programming language for developing the application. Its simplicity, readability, and extensive standard libraries make it an ideal choice for rapid application development.

II. **Tkinter:**
Tkinter is the standard GUI (Graphical User Interface) toolkit that comes with Python. It provides the necessary components for creating the application's graphical interface, including frames, labels, buttons, entry fields, and list boxes.

III. **Integrated Development Environment (IDE):**
An integrated development environment, such as PyCharm, Visual Studio Code, or IDLE, is used for coding, debugging, and testing the Python scripts.

Chapter 2

2. Features & Functionality

Start and Stop Recording Buttons:

The application includes two buttons, "Start Recording" and "Stop Recording," which initiate and terminate the screen recording process, respectively.

Threaded Recording:

To prevent the graphical user interface from becoming unresponsive during recording, the screen capture process is executed in a separate thread. This ensures a smooth user experience.

Video Format:

The captured frames are encoded using the XVID codec and saved in AVI format. Consideration could be given to using more modern and widely supported formats such as MP4. The functionality of the screen recording tool revolves around providing users with a straightforward and efficient means to capture their on-screen activities. The following features outline the core functionality of the project:

User-Friendly Interface:

Start Recording Button: Initiates the screen recording process.

Stop Recording Button: Terminates the ongoing recording session.

Graphical User Interface (GUI): Utilizes Tkinter to create an intuitive and visually appealing interface.

Threaded Execution:

Separate Recording Thread: Executes the screen capture process in a separate thread to maintain GUI responsiveness during recording.

Screen Capture and Conversion:

PyAutoGUI for Screenshots: Utilizes PyAutoGUI to capture screenshots of the entire screen.

OpenCV for Frame Conversion: Converts captured screenshots to video frames using the OpenCV library.

Video Encoding and Saving:

VideoWriter for Encoding: Employs OpenCV's VideoWriter to encode and save the captured frames as a video file.

Exception Handling:

Chapter 3

3. Working of Project

Initialization:

The program begins by importing necessary libraries, including cv2 (OpenCV), pyautogui for capturing screenshots, tkinter for the graphical user interface, and threading for executing the recording process in a separate thread.

An instance of the ScreenRecorder class is created, which initializes the Tkinter GUI with "Start Recording" and "Stop Recording" buttons.

Recording Initialization:

When the user clicks the "Start Recording" button, the start_recording method is triggered.

The method sets the recording flag to True and configures the button states accordingly.

It also initializes the VideoWriter object from OpenCV to write the frames to a video file ("screen_record.avi" in this case).

A separate thread (record_screen method) is started to handle the screen recording process.

Recording Thread:

The record_screen method continuously captures screenshots using pyautogui.screenshot() in a loop while the recording flag is True.

Each screenshot is converted to a video frame by converting the image to a NumPy array and changing the color space to BGR using OpenCV.

The resulting video frames are written to the VideoWriter object, effectively creating a video stream.

GUI Responsiveness:

To maintain GUI responsiveness during recording, the record_screen method runs in a separate thread, preventing the main thread (Tkinter GUI) from freezing.

The self.root.after(1, self.root.update) line is included to update the Tkinter GUI from the recording thread.

Stopping the Recording:

When the user clicks the "Stop Recording" button, the stop_recording method is called.

It sets the recording flag to False, waits for the recording thread to finish using self.record_thread.join(), and releases the resources (closes the VideoWriter object). Button states are adjusted to allow the user to start a new recording.

Exception Handling:

The record_screen method includes a try-except block to handle the pyautogui.FailSafeException that may occur if the mouse is moved to the upper-left corner, triggering PyAutoGUI's failsafe feature.

GUI Interaction:

The Tkinter GUI provides visual feedback to the user by enabling/disabling the recording buttons based on the recording state.

The user interacts with the GUI by clicking the "Start Recording" and "Stop Recording" buttons.

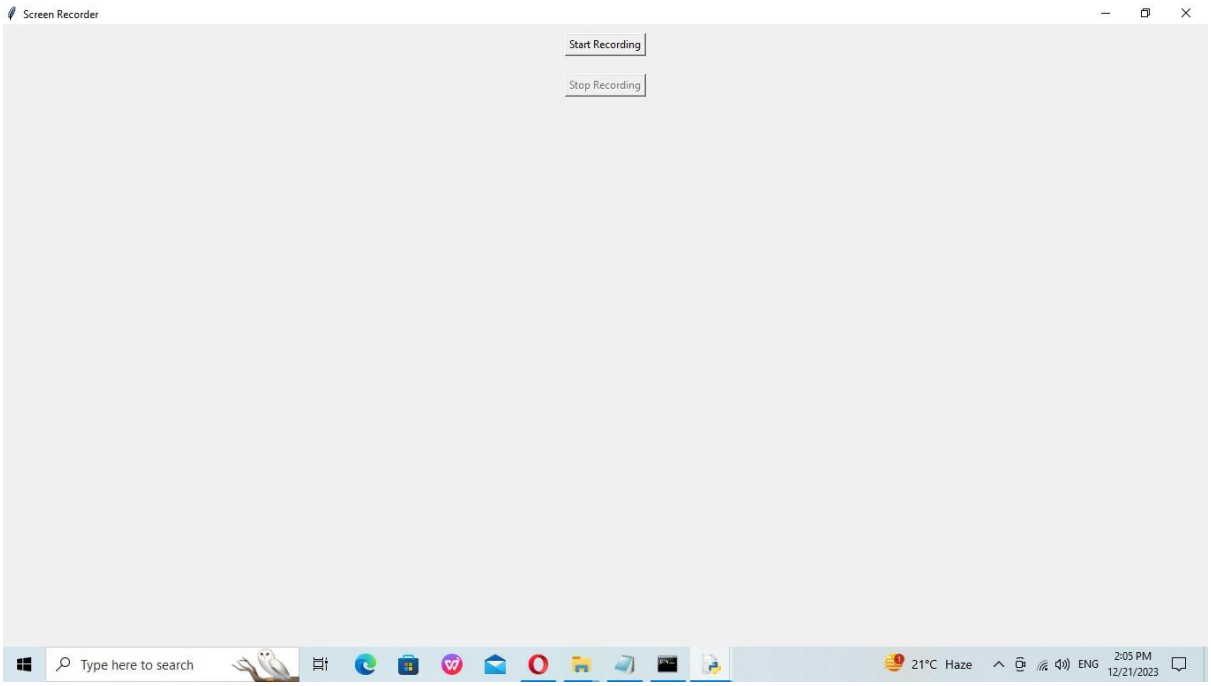
Usability and Future Improvements:

The project is designed to be accessible to users with varying technical proficiency, providing a simple solution for screen recording.

Chapter 4

4. Result/Output

App Interface



Chapter 5

5. Conclusion and Future Scope

5.1 Conclusion

The Simple Screen Recorder project provides a basic yet functional screen recording application using Python. It serves as a starting point for further enhancements and improvements to meet specific user needs. The incorporation of additional features and improvements will contribute to a more versatile and user-friendly screen recording tool.

Future Scope

Recording Duration Display: Add a feature to display the duration of the recording in the GUI.

User-Defined Output File: Allow users to specify the output file name and location.

Recording Region Selection: Implement the option for users to choose the specific region of the screen to be recorded.

Reference

1. Tkinter Documentation: https://docs.python.org/3/library/tkinter.html
2. https://edube.org/learn/pcpp1-4-gui-programming/python-professional-course-series-gui-programming