Team Ninja:

- ANIKETH
- AVINASH
- VIKAS
- GANGADHAR
- SHREYANSH

Project Report: Typing Speed Test Application

1. Introduction

This report outlines the development of a custom Typing Speed Test application built using Python's Tkinter library. The goal was to create a simple, functional tool that allows users to measure their typing speed and accuracy against a clock. It features a countdown timer, real-time input monitoring, and a final performance summary.

2. How It Works

The concept is straightforward: the app opens a window displaying a target paragraph, a text entry box, and a "Start" button. Once the user begins, a 30-second timer counts down. The user attempts to replicate the text exactly. When the time runs out—or if the user finishes early—the program stops and calculates the Words Per Minute (WPM) and accuracy percentage.

3. Under the Hood

To make this work, we relied on a few key components:

Libraries: We used tkinter to build the graphical interface and the time module to handle the countdown logic.

State Management: The app tracks the state of the game using variables for the target text, the start timestamp, a "running" flag to prevent multiple starts, and the constant 30second time limit.

4. Key Functions

Here is how the logic flows through the main functions: start_test(): This hits the reset button. It clears the input field, resets the timer, removes old scores, and initializes the countdown.

check_text(): This is the listener. It runs every time a key is released to check if the user's input matches the target text. If they type the whole paragraph correctly before time is up, this function triggers the end of the test. update_timer(): This is the heartbeat of the app. It refreshes every second to update the countdown label. Once it hits zero, it forces the test to end.

finish_test(): The scorekeeper. It compares the user input against the original text, counts the correct words, calculates the WPM and accuracy, and locks the input field so no more changes can be made.

5. User Interface

We kept the design clean and beginner-friendly. Using standard Tkinter widgets (Labels, Entries, and Buttons) stacked vertically, the layout guides the user naturally from the instructions down to the input area and results.

6. What Went Well

Simplicity: The code is lightweight and easy to follow, making it a great case study for learning Tkinter.

Portability: It runs on standard Python installations without needing heavy external libraries.

Logic: The automatic timer and word-by-word checking work seamlessly in the background.

7. Challenges Faced

Developing this wasn't without its hurdles:

Unforgiving Logic: The string matching is very strict. One missed space or typo shifts the whole string, causing the program to mark subsequent correct words as wrong.

Input Limitations: Using a single-line Entry widget makes typing long paragraphs feel cramped.

Visual Feedback: Currently, the user types "blindly" regarding errors; the app doesn't highlight mistakes in red while they are typing, which makes self-correction difficult.

8. Future Improvements

If we were to expand this project, we would focus on:

Real-time Feedback: highlighting typos in red and correct words in green as the user types.

Better Input: Switching to a multi-line text box for a better writing experience.

Variety: Randomizing the target paragraphs so users don't memorize the text.

Gamification: Adding sound effects, high scores, or difficulty levels to make it more engaging.

9. Conclusion

Overall, building this Typing Speed Test was a practical way to understand event handling and GUI design in Python. While the current version is a functional prototype, there is plenty of room to polish the user experience and add more advanced features in the future.

10.Linkdin post link:- https://www.linkedin.com/posts/jadhav-aniketh-a0716b1aa_python-tkinter-firstproject-activity-7404105234970812417-ZoT7?utm_source=share&utm_medium=member_desktop&rcm=ACoAADDKIecBIbugYLgd_HuFJxPjUAjAn0GBw68