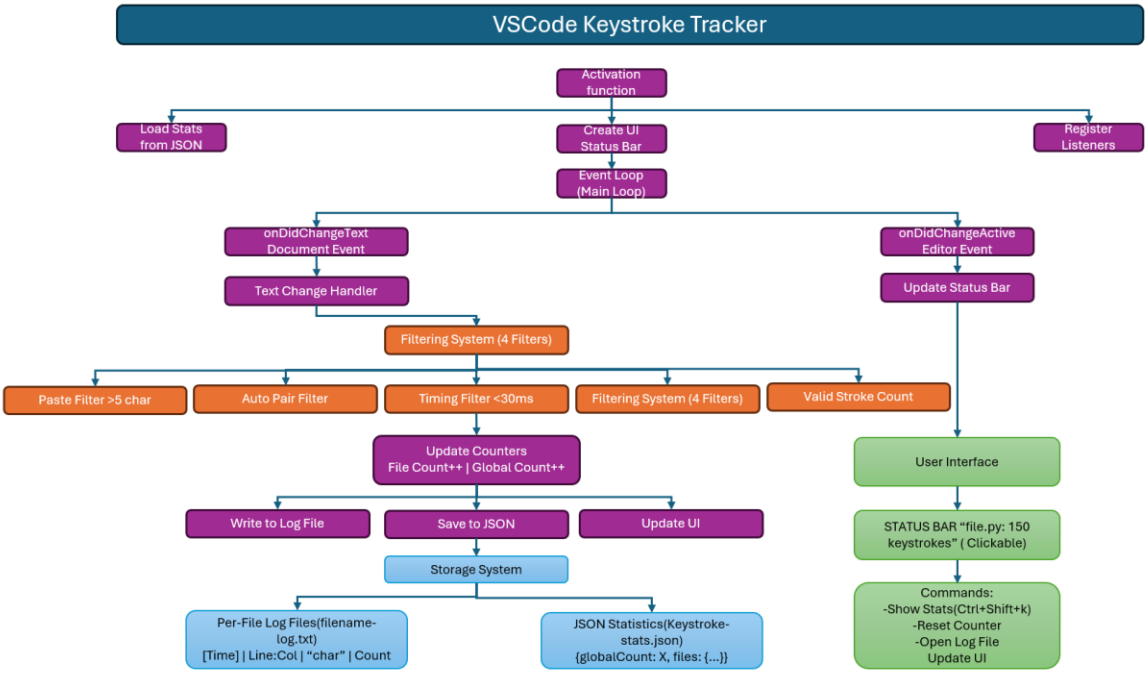
# KEYSTROKE TRACKER EXTENSION FOR VSCODE

**Author: Dr. Yongkai Wu, Akshay Mhetre**

**Abstract:** The Keystroke Tracker is a Visual Studio Code extension designed to monitor and log every manual keystroke made by developers while coding, providing real-time tracking, persistent storage, and per-file logging capabilities for analyzing coding patterns and productivity metrics. While the extension successfully filters out auto-complete, paste operations, and automated text insertions through heuristic-based detection, it has limitations including API constraints in distinguishing all manual versus programmatic inputs, support for only common code file extensions, and lack of deletion tracking. This tool benefits students by providing productivity analysis, assignment verification proving manual work rather than copied code, and self-assessment of coding habits, while companies gain developer productivity metrics, improved project estimation through historical keystroke data, code quality insights from keystroke complexity patterns, onboarding progress tracking, and compliance records for development activity auditing all achieved through non-intrusive monitoring that respects developer privacy by focusing on keystroke counts rather than capturing actual code content.

**Objective:** The primary objectives of this research project were to develop and implement a Visual Studio Code extension capable of accurately quantifying manually typed keystrokes through intelligent filtering mechanisms, establish a robust persistent logging system that maintains individual records for each file while ensuring data integrity across multiple IDE sessions, design and deploy automated filtering algorithms to distinguish between user-generated input and system-generated text such as auto-complete suggestions and clipboard paste operations, integrate real-time visual feedback mechanisms through status bar instrumentation to provide continuous keystroke count updates during active development sessions, and engineer a user-centric installation framework that minimizes technical barriers to entry by enabling deployment through a streamlined single-file download and installation workflow accessible to non-technical users.

**System Architecture:**




**Implementation:**

The Keystroke Tracker extension works in three main stages: starting up, running, and shutting down. When you open VSCode, the extension activates by loading any saved keystroke counts from files, creating the counter display you see at the bottom of the screen, and setting up listeners to watch what you type. While you code, the extension constantly monitors your typing, filters out anything that isn't a real keystroke like copy-paste or auto-complete, counts only the characters you actually type, saves everything to both a detailed log file and a summary file, and updates the display showing your current count. When you close VSCode, the extension saves all your counts one final time so nothing is lost.

The extension has five key parts working together. First, the startup system loads your previous counts and prepares everything. Second, the typing monitor watches every change you make to your files. Third, the smart filter checks each change using four tests: it ignores big chunks of text over five characters assuming they're pasted, recognizes when VSCode adds closing brackets or quotes and only counts your opening one, checks if multiple characters appear within thirty milliseconds which means auto-complete, and makes sure you're typing in actual code files not the log files. Fourth, the storage system keeps two types of records: a summary file listing counts for all your files and individual detailed logs for each file showing every character you typed with timestamps. Fifth, the display system shows

your current file's count at the bottom of VSCode and provides commands to view statistics, reset counts, or open log files.

The extension is built using TypeScript code that gets converted to JavaScript, runs inside VSCode using their extension system, reads and writes files using standard file operations, and gets packaged into a single installable file. The counting logic is straightforward: when text changes, first check if it's a deletion and skip it, then check if it's more than five characters and skip it as a paste, then check if it's bracket or quote pairs and count only one, then check if it's a single character and count it, then check if multiple characters appeared too fast and skip them as auto-complete, otherwise reject it as automated input. The display updates by finding which file you're currently editing, looking up how many keystrokes that file has, and showing it as "filename.py: 150 keystrokes" at the bottom of your screen.

**Tests:** Various comprehensive tests were performed to validate the extension's functionality and accuracy. The testing process for the Keystroke Tracker extension was organized into five comprehensive categories to ensure reliability and functionality across different use cases. Functional testing verified the core features of the extension including the accuracy of manual keystroke counting, proper detection of auto-complete pairs like brackets and quotes, effective filtering of copy-paste operations, correct tracking across multiple files simultaneously, verification that data persists properly in storage files, and confirmation that the status bar updates accurately in real-time. Edge case testing examined how the extension handles unusual or challenging scenarios such as rapid typing where keystrokes occur in quick succession, situations mixing manual typing with auto-complete suggestions, proper handling of special characters and newline characters, and behavior when working with empty files. Integration testing focused on how well the extension works within the VSCode environment, checking that activation and deactivation occur smoothly when VSCode starts and stops, verifying correct behavior when users switch between different files, ensuring all commands execute properly when triggered, and confirming that log files are created correctly and new entries are appended without errors. User acceptance testing evaluated the extension in realistic conditions by testing it during actual coding sessions, verifying functionality across multiple programming languages including Python, JavaScript, and C++, and observing performance with various coding patterns and styles that developers commonly use. Performance testing measured the extension's efficiency by recording response times to individual keystrokes to ensure no noticeable lag, and evaluating file input and output performance to confirm that reading and writing operations do not slow down the editor or interfere with normal coding workflow.

**Results:** The Keystroke Tracker extension successfully accomplishes several important goals. First, it accurately counts only the keystrokes you actually type by filtering out auto-

complete suggestions, copy-paste operations, and other automated text that VSCode adds, achieving high accuracy in telling the difference between what you manually typed versus what the computer generated automatically. Second, all your keystroke counts are saved permanently so they don't disappear when you close VSCode, with each file maintaining its own separate count that persists across sessions using a reliable file-based storage system. Third, the extension is easy to use with a live counter displayed at the bottom of your screen that updates as you type, simple commands you can run with keyboard shortcuts, and log files you can easily open and view anytime. Fourth, it keeps detailed records of every character you type including exactly when you typed it, which line and column it was on, and organizes all this information into separate log files for each code file you work on. Finally, the extension is extremely lightweight, using less than 2 megabytes of your computer's memory, which means it runs smoothly without slowing down VSCode or your computer.

**Future Enhancements:**

Potential improvements for future versions:

1. Advanced Analytics
    - Typing speed calculation
    - Productivity graphs and trends
    - Time-of-day analysis
    - Weekly/monthly reports
2. Cloud Sync
    - Synchronize statistics across devices
    - Team-wide analytics
    - Centralized reporting
3. Integration
    - Export to CSV/Excel
    - API for external tools
    - Integration with project management systems
4. Privacy Controls
    - Anonymous mode (no character logging)

    - Selective file tracking
    - Data encryption options
5. Multi-language Support
    - Internationalized UI
    - Localized documentation

**Conclusion:** The Keystroke Tracker extension proves that we can learn valuable information about how developers work by simply watching what they type, without being invasive or intrusive. Instead of using old-fashioned measurements like counting lines of code, this extension focuses on actual manual typing, which gives us a better and more honest picture of how much real effort a developer puts into their work. The extension is fully ready to use in real-world situations, whether you're a student coding alone or a company wanting to track team productivity. Because it's built with a clear, organized structure and comes with detailed instructions, anyone can easily modify it or add new features in the future to fit their specific needs.

One aspect to note is that the extension's behavior varies slightly depending on how files are opened. When opening files through VSCode's "File" menu and then "Open File..." option, the extension correctly loads and displays all previous keystroke counts. However, when files are opened by double-clicking them directly from the file system, the saved count is not displayed in the status bar. This is related to VSCode's file activation event handling and represents an area for potential refinement in future versions.

**References**

**VSCode Extensions:**


- WakaTime: https://wakatime.com/github-time-tracking
- Code Time: https://marketplace.visualstudio.com/items?itemName=softwaredotcom.swdc-vscode


**GitHub Projects:**


- Browser based typing game: https://github.com/knadh/wordpluck


**Learning Resources**

**TypeScript:**


- TypeScript Handbook: https://www.typescriptlang.org/docs/handbook/intro.html

- Node.js File System: https://nodejs.org/api/fs.html

**VSCode Extension Development:**

- Extension Guides: https://code.visualstudio.com/api/extension-guides/overview
- Testing Extensions: https://code.visualstudio.com/api/working-with-extensions/testing-extension

**Communities:**

- VSCode Extensions Discord: https://aka.ms/vscode-dev-community
- Stack Overflow - VSCode Extension: https://stackoverflow.com/questions/tagged/vscode-extensions