ShortKey: The Ultimate Keyboard Shortcut Trainer

1 Introduction

ShortKey is an innovative Python application designed to help users learn and master keyboard shortcuts through real-time monitoring and feedback. The application intercepts keyboard input while allowing normal keyboard usage, providing an interactive learning experience directly in the terminal.

2 Discovery Project Idea

The initial concept for ShortKey emerged from observing the inefficiency of using the mouse to navigate the desktop. The pitch focused on creating a real-time learning device that would:

- Provide immediate feedback on keyboard shortcut usage
- Help users discover new shortcuts they might not know
- Create a more interactive learning experience than traditional documentation
- Support both system-wide and application-specific shortcuts

3 Project Progress

The project evolved through several key phases:

3.1 Initial Brainstorming

Originally, I had planned for ShortKey to be a device that would emulate a keyboard. However, after some research and testing, I realized that it would be too difficult to emulate a keyboard with the Raspberry Pi Model 5. As an alternative, I decided to make a software based solution that would suggest shortcuts to the user in their terminal.

3.2 Initial Development

Once I had the idea for the project, I began to work on the code. I began to research how to intercept keyboard input in Linux. I found a python library called evdev that would allow me to intercept keyboard input without removing it from the host system. In order to make the data recieved from evdev useful, I emplemented a hashmap/dictionary to map the keyboard input to plain text in order to search the shortcut file I had planned on creating.

3.3 Shortcut File

To make the shortcut file, I tasked Claude to make a JSON file that would contain an exhaustive list of keyboard shortcuts for Windows/Linux. Once made, I added the JSON file to the project and began to work on the code to make the program functional.

3.4 Debugging

When I first started the program and began to test it, I noticed that the program was not recieving any input from my keyboard. After some debugging, I realized that my keyboard was special and was showing up as three different devices. In order to fix this, I had to add a check to the program to use the event0 device instead of whichever device was presented to the program first.

3.5 Adding Functionality

After this was fixed, I began to work on the code to make the program functional. I added a function that would check if a key was pressed and then check if any of the shortcuts in the JSON file matched the key pressed. If matches were found, the program would suggest shortcuts to the user. If a valid shortcut was completed, the program would notify the user and describe the shortcut's function.

3.6 Refinement

I then added support for Chrome shortcuts and a browser mode that would allow the user to use the shortcuts in their web browser. With this, I implemented a 'layer' system that would allow the user to toggle browser mode on and off.

3.7 Initial Product

- Basic keyboard input monitoring using evdev
- Simple shortcut recognition system
- Terminal-based interface implementation

3.8 Future Development

- Optimization of shortcut recognition algorithms
- Enhancement of user feedback mechanisms
- Documentation and code organization improvements

4 Project Successes and Failures

4.1 Successes

- Successful implementation of real-time keyboard monitoring
- Creation of a flexible shortcut configuration system
- Development of an intuitive browser mode
- Effective prediction system for shortcut suggestions

4.2 Failures

- The program was not able to send outbound HID signals to another machine.

5 ECE Skills Gained

Through the development of ShortKey, several key ECE skills were developed and enhanced:

5.1 Technical Skills

- System-level programming with Python
- SSH protocol
- Using Efficient Data Structures to improve program performance
- Input device handling and event processing
- Modular software design
- Debugging complex input handling scenarios
- HID signal processing

6 Key Features

The application includes several notable features:

- Real-time keyboard input interception
- Configurable shortcuts through JSON
- Special browser control mode (activated by Ctrl+Win+Alt)
- Real-time shortcut prediction and suggestions
- Terminal-based interface

7 Technical Implementation

The application is built using Python 3.x and leverages several key components:

7.1 Core Components

- `evdev` for keyboard input handling
- JSON configuration for shortcut definitions
- Modular design with separate shortcut recognition and prediction systems

7.2 Architecture

The application follows a modular architecture with:

- Keyboard input monitoring
- Shortcut recognition system
- Browser mode management
- Real-time prediction engine

8 JSON Configuration

Shortcuts are defined in a JSON file (`KeyboardShortcut.json`) with two main categories:

- System shortcuts
- Browser-specific shortcuts

Each shortcut includes:

- Description
- Action type
- Specific action

9 Usage

To use ShortKey:

1. Install dependencies: `pip install evdev`
2. Make the script executable: `chmod +x ShortKey.py`
3. Run the application: `./ShortKey.py`

10 Compatibility

The application has been tested on:

- Raspberry Pi 5
- Python 3.x
- Required packages: evdev, logging, json