Software Requirements Specification

for

CMD TypeSwift Typing Tutor

Prepared by

Athul Babu K – NIE22CS019

Sobin Seby – NIE22CS047

Sojo Jenson – NIE22CS048

DATE: 10-01-25

Table of Contents

1. Introduction

1.1 Purpose

This document specifies the software requirements for the "CMD TypeSwift Typing Tutor with Integrated Programming Lessons" software, which will be implemented using C programming. The software aims to improve the typing speed and accuracy of users while providing integrated programming lessons. It consists true programming lessons, so the user can type and learn at the same time.

1.2 Document Conventions

This document follows standard IEEE SRS conventions. Requirements are uniquely identified with a sequence number.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, marketing staff, users, testers, and documentation writers. It is organized to provide an overview of the software, followed by detailed requirements.

1.4 Product Scope

Despite training in programming and software development, many computer science engineering graduates often lack proficient typing skills and speed. This gap can be attributed to the focus on coding logic and software architecture rather than also including basic typing proficiency in their curriculum. As a result, even though they have advanced technical knowledge, their efficiency in writing code and documentation can be hindered by slow and inaccurate typing. This highlights the need for integrating typing skill development into computer science education to enhance overall productivity. This project aims to develop an interactive software application that combines typing practice with programming lessons, providing users with a dual advantage of improving typing speed and learning coding simultaneously.

1.5 References

IEEE SRS Template

Github:

typing-tutor · GitHub Topics · GitHub

https://github.com/Bullet4Men/Speed-Typing-Test

https://stackoverflow.com/questions/19875136/continuous-keyboard-input-in-c

2. Overall Description

2.1 Product Perspective

This software is a standalone application designed to run on command-line interfaces. It is not part of a larger system but can be integrated with other educational tools.

2.2 Product Functions

- Typing practice exercises
- Typing speed and accuracy tracking
- Integrated programming lessons
- Progress tracking and reporting

2.3 User Classes and Characteristics

- Beginners: Users with basic typing skills and no programming knowledge.
- Intermediate: Users with moderate typing skills and some programming knowledge.
- Advanced: Users with high typing proficiency and advanced programming knowledge.

2.4 Operating Environment

CMD TypeSwift will be a programmer-oriented typing tutor tutor designed for Linux distros written in C, designed to work on the gnome-terminal, xterm and uxterm and other consoles.

2.5 Design and Implementation Constraints

- Implemented using C programming language
- Command-Line interface only
- Users must navigate through the terminal interface

2.6 User Documentation

- **User manual** containing:

Installation guide, Run guide, Command prompt arguments list, Uninstall guide.

2.7 Assumptions and Dependencies

- Users of CMD TypeSwift are expected to have access to a command-line interface like gnome-terminal, xterm and uxterm.
- Users have basic knowledge of operating a command-line interface.

3. External Interface Requirements

3.1 User Interfaces

CMD TypeSwift will use a command-line interface with text-based menus and prompts as the user interface. Terminal will be manipulated to print green a correct character and red a wrong character. This will make CMD TypeSwift command line tutor very appealing. The user easily knows where they have gone wrong and can undo all errors. User session timing starts only when the user types a first character (wrong or correct). The last line of each lesson session will be marked blue, so user is aware and can take time to review speed details before starting the following lesson.

3.2 Hardware Interfaces

Keyboard for typing input – CMD TypeSwift supports both DVORAK and QWERTY keyboard styles.

3.3 Software Interfaces

CMD TypeSwift will use standard C libraries for input/output operations; and custom libraries for basic functions needed, name and session display management. File manipulations, Terminal functions to manipulate console.

4. System Features

4.1 Typing Lessons

4.1.1 Description and Priority

Provides a comprehensive set of typing lessons for both non-programmers (basic techniques) and programmers (advanced techniques). This feature is of High priority due to its critical role in user training and skill development.

4.1.2 Stimulus/Response Sequences

- User Action: User selects a lesson.
- System Response: System displays the chosen lesson and starts tracking typing speed and errors.

4.1.3 Functional Requirements

- REQ-1: The system must provide a variety of lessons tailored to non-programmers and programmers.
- REQ-2: The system should start a new session when a lesson is selected.
- REQ-3: The system must track typing speed and errors during the session.

4.2 Speed Algorithm

4.2.1 Description and Priority

Implements an accurate speed algorithm to measure characters per minute (CPM) and an approximate words per minute (WPM) with ±0.35 accuracy. This feature is of High priority due to its importance in performance tracking.

4.2.2 Stimulus/Response Sequences

- User Action: User completes a typing session.
- System Response: System calculates and displays typing speed in CPM and WPM.

4.2.3 Functional Requirements

- REQ-1: The system must accurately measure typing speed in CPM and WPM.

- REQ-2: The system should display the calculated speed at the end of each session.

4.3 Visual Feedback

4.3.1 Description and Priority

Manipulates the terminal to print correct characters in green and wrong characters in red, providing immediate visual feedback. This feature is of medium priority due to its role in enhancing user experience and error correction.

4.3.2 Stimulus/Response Sequences

- User Action: User types a character.
- System Response: System displays the character in green if correct, or in red if incorrect.

4.3.3 Functional Requirements

- REQ-1: The system must display correct characters in green.
- REQ-2: The system must display incorrect characters in red.

4.4 Performance Tracking

4.4.1 Description and Priority

Keeps a detailed record of typing sessions, including the number of errors, speed in CPM and WPM, and time elapsed. This feature is of High priority due to its importance in tracking user progress.

4.4.2 Stimulus/Response Sequences

- User Action: User completes a typing session.
- System Response: System records the session details in a text file.

4.4.3 Functional Requirements

- REQ-1: The system must record the number of errors, speed in CPM and WPM, and time elapsed for each session.
- REQ-2: The system should store session details in a text file.

4.5 Custom Text Input

4.5.1 Description and Priority

Allows users to input their own text in the "myown.txt" file for typing practice. This feature is of Low priority due to its optional nature.

4.5.2 Stimulus/Response Sequences

- User Action: User inputs custom text in the "myown.txt" file.
- System Response: System allows the user to practice typing the custom text.

4.5.3 Functional Requirements

- REQ-1: The system must allow users to input custom text in the "myown.txt" file.
- REQ-2: The system should provide a standard mode for typing the custom text.

4.6 Command Line Features

4.6.1 Description and Priority

Provides various command-line features for user convenience and customization. This feature is of High priority due to its impact on user experience and flexibility.

4.6.2 Stimulus/Response Sequences

- User Action: User chooses a typing mode (random/standard).
- System Response: System starts the session in the selected mode.
- User Action: User modifies a lesson in the lesson file.
- System Response: System updates the lesson file.
- User Action: User inputs text in the "myown.txt" file.
- System Response: System allows typing practice with the custom text.
- User Action: User types a character.
- System Response: System may produce a beep for wrong characters.

- User Action: User erases characters.
- System Response: System decrements the number of errors and resets timing if erased to the beginning.

4.6.3 Functional Requirements

- **REQ-1**: The system must allow users to choose between random mode and standard mode.
- **REQ-2**: The system should start a session based on the selected mode.
- **REQ-3**: The system must allow users to modify any lesson in the lesson file.
- **REQ-4**: The system should provide a "myown.txt" file for custom text input.
- **REQ-5**: The system must ignore extended ASCII characters and handle only 7-bit ASCII characters.
- **REQ-6**: The system should produce a beep for wrong characters (if supported by the terminal).
- **REQ-7**: The system must remember all past settings.
- **REQ-8**: The system should allow users to erase characters and adjust the number of errors accordingly.
- **REQ-9**: The system must reset session timing when the user erases to the beginning of the session.

4.7 Lesson Marking

4.7.1 Description and Priority

Marks the last line of each lesson session in blue, allowing the user to review speed details before proceeding. This feature is of Low priority due to its visual nature.

4.7.2 Stimulus/Response Sequences

- User Action: User reaches the last line of a lesson session.
- System Response: System marks the line in blue for review.

4.7.3 Functional Requirements

- REQ-1: The system must mark the last line of each lesson session in blue.
- REQ-2: The system should allow users to review speed details before starting the next lesson.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system must provide real-time typing speed and accuracy tracking. It should be able to handle any typing speed of users without any performance issues, and the response time for feedback should be quick.

5.2 Safety Requirements

No specific safety requirements are needed but the system must comply with all relevant safety regulations.

5.3 Security Requirements

Security of the progress files must be ensured.

5.4 Software Quality Attributes

The system should be adaptable, compatible with different command-line interfaces, and have 99.9% availability. It must provide accurate tracking of typing speed and errors and offer customizable lessons and typing modes. Interoperability with various Linux distributions and terminal emulators is necessary. The codebase should be well-documented and modular for maintainability, easily portable to different Linux environments, and provide consistent performance. Reusability of software components, handling user errors gracefully, being a command-line interface program, having an intuitive user interface which does not seem complicated for the users are also important.

5.5 Business Rules

User authentication may be implemented for session data access, with activity logging for auditing purposes. There should be different access levels for administrators and regular users.

6. Other Requirements

- **Database Requirements**: The system should use a lightweight, file-based database to store user progress and session data.
- **Reuse Objectives**: The software should be designed with modularity in mind, allowing for the reuse of components in other educational tools or typing tutors. This includes creating well-documented and reusable code libraries.

Appendix A: Glossary

- **CMD**: Command Line Interface
- **CPM**: Characters Per Minute
- **WPM**: Words Per Minute
- **DVORAK**: A keyboard layout designed to increase typing speed and efficiency
- **QWERTY**: The standard keyboard layout used in most English-speaking countries

Appendix B: To Be Determined List

1. TBD-1: Define the specific modular components that can be reused in other projects.
2. TBD-2: Identify ways of making the product available for multiple users from a single machine.
3. TBD-3: Identify additional accessibility features.
4. TBD-4: Plan to implement this project as a Docker image so as to make it easily available for the masses.