Touch Type Workout

Analysis Document: Software Requirement Specification

Author: Kyrylo Krysko

CONTENTS

1. INTRODUCTION

1.1. PURPOSE

The purpose of this document is to outline detailed functional and non-functional requirements for the Touch Typing Workout application. This will serve as a definitive guide for the developers, stakeholders, and users to understand the design, features, and working of the system.

1.2. SCOPE

The application, titled 'Touch Typing Workout', is intended to enhance the typing skills of users by providing a structured set of lessons and tests to practice and monitor progress. The application will be developed using Python and intended for use on the Windows operating system. It will allow users to improve their typing speed and accuracy, aiming to reach a goal of 40 words per minute.

1.3. OVERVIEW

The rest of this document will detail the specific functional and non-functional requirements, system features, constraints, and system interactions that are essential for the successful development and operation of the Touch Type Workout application

2. GENERAL DESCRIPTION

2.1. PRODUCT PERSPECTIVE

The TTW application will be a standalone application meant to run on Windows 10 operating system. It is a training and testing tool designed for improving touch typing skills.

2.2. PRODUCT FUNCTIONS

A general overview of the main functionality includes:

- Providing a set of structured lessons and tests for users to improve their touch typing skills.
- Displaying typing tasks for users in the form of text (what keys to press).
- Visual feedback on a keyboard image (what keys were pressed).
- Providing progress tracking with metrics such as accuracy, speed, and rhythm of typing.

2.3. USER CHARACTERISTICS

The users of the TTW application are expected to be individuals wishing to improve their typing skills. As such, they should have a basic understanding of how to use a computer and keyboard.

3. FUNCTIONAL REQUIREMENTS

3.1. LESSONS AND TESTS

The application should provide a structured set of lessons and tests for touch typing practice. Each lesson should focus on different groups of keys, starting with the home row keys and gradually incorporating upper and lower row keys. Lessons should progressively increase in complexity, starting from single letters to words and then sentences. Tests should simulate real-world typing scenarios with varying lengths of paragraphs. After each lesson or test, users should receive detailed feedback on their performance including typing speed, accuracy, and rhythm.

3.1.1 LESSONS DATA

Lessons data included in Appendix 6.1

3.1.2 TESTS DATA

Lessons data included in Appendix 6.2

3.2. TYPING TASKS DISPLAY

Each lesson or test should be displayed as a text task, clearly indicating which keys to press. The text task should be dynamically updated as the user types, highlighting the next key to be pressed.

3.3. VISUAL FEEDBACK

The application should provide visual feedback on a keyboard image, highlighting the keys that were pressed within 50 milliseconds. This visual feedback should be differentiated for correctly and incorrectly pressed keys - correctly pressed keys can be highlighted in green while incorrect ones can be highlighted in red.

3.4. PROGRESS TRACKING

Users should be able to track their progress over time. This progress should be calculated based on the speed, accuracy, and rhythm of typing in each lesson or test. It should be displayed in a graphical form - as a line graph with time on the X-axis and speed, accuracy, and rhythm on the Y-axis.

3.5. USER INTERFACE

The user interface (UI) for the TTW application is an essential part of the functional requirements and should provide an interactive, intuitive, and user-friendly experience.

The UI should include the following components:

- **Login/Registration Page**: This is the first page users will interact with. It should contain fields for username and password entry for returning users, as well as a button to redirect to a registration page for new users.
- **Main Dashboard**: After successful login, users will be redirected to a main dashboard. This dashboard should contain buttons or links for accessing the lessons and tests, viewing progress reports, changing account settings, and logging out of the application.
- **Lessons and Tests Page**: This page should contain a list of all the available lessons and tests, including those the user has already completed. Users should be able to select a lesson or test to start, pause, or resume it.
- **Typing Task Display**: During a lesson or test, the typing task should be displayed prominently at the top of the page. The keys that need to be pressed should be clearly indicated.
- **Keyboard Visualization**: Below the typing task, there should be a visual representation of a keyboard. This visualization should highlight the keys as they are pressed by the user.
- **Progress Page**: This page should display a graph or charts showing the user's progress over time. It should include metrics such as typing speed, accuracy, and rhythm.
- **Settings Page**: This page should allow users to change their account settings, such as username, password, and email.

Overall, the UI should maintain a consistent theme across all pages. It should utilize clear and legible fonts, contrasting colors for better visibility, and easy-to-understand icons and tooltips. All interactions should provide immediate feedback to the user, such as a change in button color when clicked. The UI should also be designed to accommodate different screen sizes and resolutions to cater to various devices.

4. NON-FUNCTIONAL REQUIREMENTS

4.1. PERFORMANCE REQUIREMENTS

The application should be responsive and fast, providing immediate feedback to the user's inputs.

- **Response Time**: The application should be highly responsive to user inputs. The user interface should respond to user actions (like mouse clicks or key presses) within 100 milliseconds.
- **Feedback Time**: Visual feedback on the keyboard image highlighting the keys that were pressed should be provided within 50 milliseconds after key press to ensure a real-time experience.

- **Loading Time**: The application should load within 2 seconds to ensure that users do not have to wait too long. Each lesson or test should load within 1 second after being selected.
- **Processing Time**: The application should process user performance data and update the progress charts within 500 milliseconds after the completion of each lesson or test to provide immediate feedback to the users.

4.2. USABILITY REQUIREMENTS

The application should have a user-friendly interface, making it easy to navigate and use even for beginners.

4.3. SYSTEM REQUIREMENTS

The application should run on the Windows operating system. The application should be developed with Python3.7 language. Non-standard libraries allowed to be used.

The system should run on a computer with at least a dual-core processor and 2GB of RAM for smooth operation.

A Standard English keyboard is required for users to practice touch typing.

4.4. DEPLOYMENT

The application should be deployed as a package with cmd launcher and include Python 3.7 with necessary libraries.

4.4. USER REQUIREMENTS

Users should have a basic understanding of how to use a computer and keyboard. They should be able to navigate through the lessons and tests, understand the tasks and feedback provided, and interpret the progress graphics. Users should also be able to create an account and log in to track their progress over time.

5. TEST PLAN

The purpose of the Test Plan is to define the testing approach and to manage and control testing activities. This section provides the plan to verify and ensure that the Touch Typing Workout application meets its design specifications and other requirements.

5.1. TEST STRATEGY

The application will be tested using the following strategies:

- **Unit Testing**: Each component will be individually tested to ensure it works correctly in isolation.
- **Integration Testing**: The components will then be combined and tested together to identify issues in the interaction between the different components.
- **System Testing**: The entire system will be tested in an environment that simulates the final production environment to identify any system-level issues.
- **User Acceptance Testing**: The application will be tested in a real-world situation by end-users to ensure that it meets the users' needs and can perform the tasks it was designed to perform.

5.2. TEST SCENARIOS

Some of the key test scenarios that will be used to validate the application are:

- **Test Lessons and Tests**: Verify that the lessons load properly, tasks display as expected, and user input is correctly captured and evaluated.

- **Test Visual Feedback**: Confirm that the application provides correct visual feedback on the keyboard image, highlighting the keys that were pressed.

- **Test Progress Tracking**: Check that the user's progress is correctly recorded and displayed, and that changes over time are correctly reflected.

- **Test System Performance**: Ensure that the application responds quickly to user inputs, that the visual feedback and progress updates are almost instantaneous, and that the application loads quickly.

- **Test User Interface**: Confirm that the user interface is easy to navigate, all buttons, links, and menus work correctly, and the visual elements are displayed correctly on different screen sizes and resolutions.

5.3. TEST ENVIRONMENT

The test environment should be set up to closely mimic the production environment. This includes using the same type of system that the end-users are expected to use, including the hardware, operating system, and other software, and the same network conditions.

Test will be done at my Windows OS PC.

5.4. TEST DATA

The testing process will require data that reflects the kind of data that the application will handle in the real world. This includes test sample lessons and tests, and recorded user inputs. The test data should be diverse and cover as many different situations as possible.

5.6. TEST TOOLS

The testing process will be done manually due to the time constrains, we don’t have enough resources to provide a system for automated tests.

6 APPENDIXES

6.1. LESSONS DATA

- Lesson 01:
  - Letters: F, J
  - Sequences:
    - "fff jjj fff jjj fff jjj"
    - "fjf fjf fjf fjf fjf fjf"
    - "jff jff jff jff jff jff"
    - "fjj fjj fjj fjj fjj fjj"
- Lesson 02:
  - Letters: D, K
  - Sequences:
    - "ddd kkk ddd kkk ddd kkk",
    - "dkd dkd dkd dkd dkd dkd",
    - "kdd kdd kdd kdd kdd kdd",
    - "ddk ddk ddk ddk ddk ddk"

6.2. TEST DATA

- test_01:
  - Paragraphs:
    - "The quick brown fox jumps over the lazy dog."
    - "Jackdaws love my big sphinx of quartz."
    - "How vexingly quick daft zebras jump."
    - "Bright vixens jump; dozy fowl quack."