Touch Type Workout

Software Design Document

Author: Kyrylo Krysko

CONTENTS

1. INTRODUCTION

This Software Design Document provides a comprehensive architectural overview of the Touch Typing Workout application, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.1. REFERENCE DOCUMENTATION

This document based on Software Concept and Software Requirements documents

1.2. APPLICATION SCOPE

The application is intended to enhance the typing skills of users by providing a structured set of lessons and tests to practice and monitor progress. It will allow users to improve their typing speed and accuracy, aiming to reach a goal of 40 words per minute.

Major software functions:

- **Lesson Data**: Application contains a series of structured typing lessons. Each lesson is designed to incrementally increase the complexity and range of keys used. Lessons stored in JSON format and shipped with application package.
- **Test Data**: Application contains a series of typing tests for users. The tests are structured to simulate real-world typing scenarios. The user's performance on these tests is recorded and used to measure progress. Tests stored in JSON format and shipped with application package.
- **User Input Tracking**: Monitors and records the user's key presses in real-time during lessons and tests. This is used to provide immediate feedback to the user and for progress tracking.
- **Visual Feedback**: Provides real-time visual feedback to the user during lessons and tests. It displays a keyboard image and highlights the keys as they are pressed.
- **Progress Tracking**: Tracks and stores user progress over time. This is done based on key metrics such as typing speed, accuracy, and rhythm. It allows the user to see how they are improving over time. Data should be stored in JSON format in user Documents folder, no database required.
- **Performance Reporting**: Provides a comprehensive report on the user's typing performance, leveraging the tracked progress data to generate visualizations that depict improvement over time. Data should be loaded from JSON format in user Documents folder, no database required.
- **User Support and Help**: Provides a user-friendly guide or manual to aid users in understanding and using the application effectively. May also include FAQ or troubleshooting assistance.
- **Time Constrains**: list of timing in milliseconds for each critical application functions

2. ARCHITECTURAL DESIGN

This section contains high level overview of the system components.

The application follows an Object-Oriented Programming design, with the primary class, **TouchType**. The application GUI uses the **PyQt** framework. UI file made in **QTDesigner**, compiled and imported into main module.

The **TouchType** class encapsulates all application functionalities and state variables. The primary responsibilities of the **TouchType** class are:

- Initializing the user interface (UI) and loading application and user data (statistics).
- Controlling the flow of lessons and tests with build-in **keyPressEvent**() method.
- Providing user ablity to run test and lessons and observe statistics.
- Processing user inputs (keys press) and providing visual feedback.
- Recording and displaying statistics.
- Providing recommendations based on the user's previous performance.
- Utilize **matplotlib** library to draw graphs.

3. DETAILED DESIGN

This section contains detailed description of the whole application.

In the **TouchType** class:

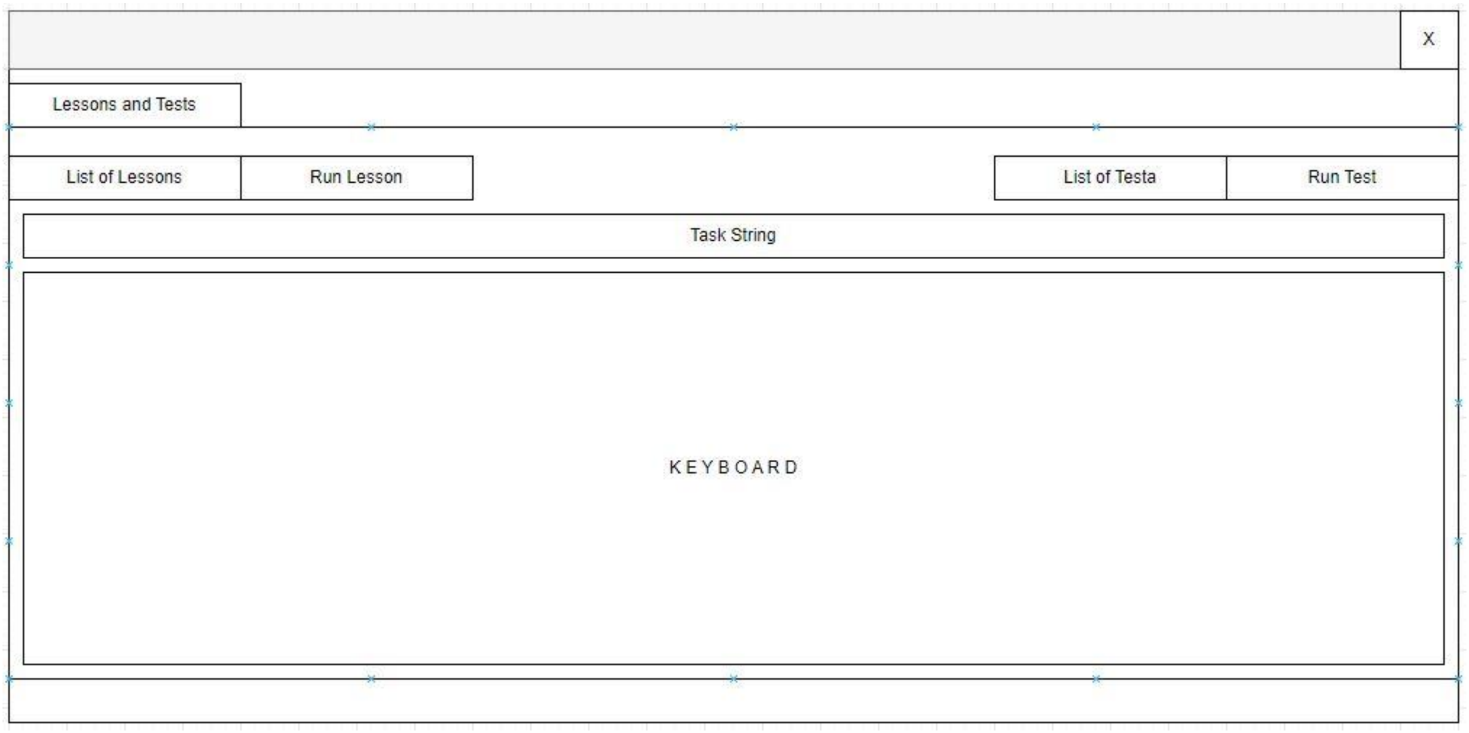
- **__init__** initializes the UI and the initial application state.
- **init_ui** sets up the UI at application launch.
- **start_lesson** and **start_test** are responsible for starting a lesson or a test. They reset necessary variables, load the sequence of strings to type, and set the UI focus for catching key presses.
- **keyPressEvent** is an event handler that reacts to user's key presses. It controls the flow of a lesson or test, checks the accuracy of user inputs, and provides visual feedback by coloring the input text.
- **check_user_input** colors user input text to provide visual feedback.
- **set_next_picture** and **start_sequence** control the sequence of characters displayed during a lesson or a test.
- **sequence_wpm, errors_rate, cps_to_wpm, rhythm,** and **record_statistics** handle the calculations and recording of statistics like words per minute, error rate, and rhythm.
- **recommendation** provides recommendations to the user based on their previous session's statistics. It determines areas the user needs to improve and suggests methods to do so.
- **reset_ui** resets the user interface at the end of a lesson or test.

4. USER INTERFACE DESIGN

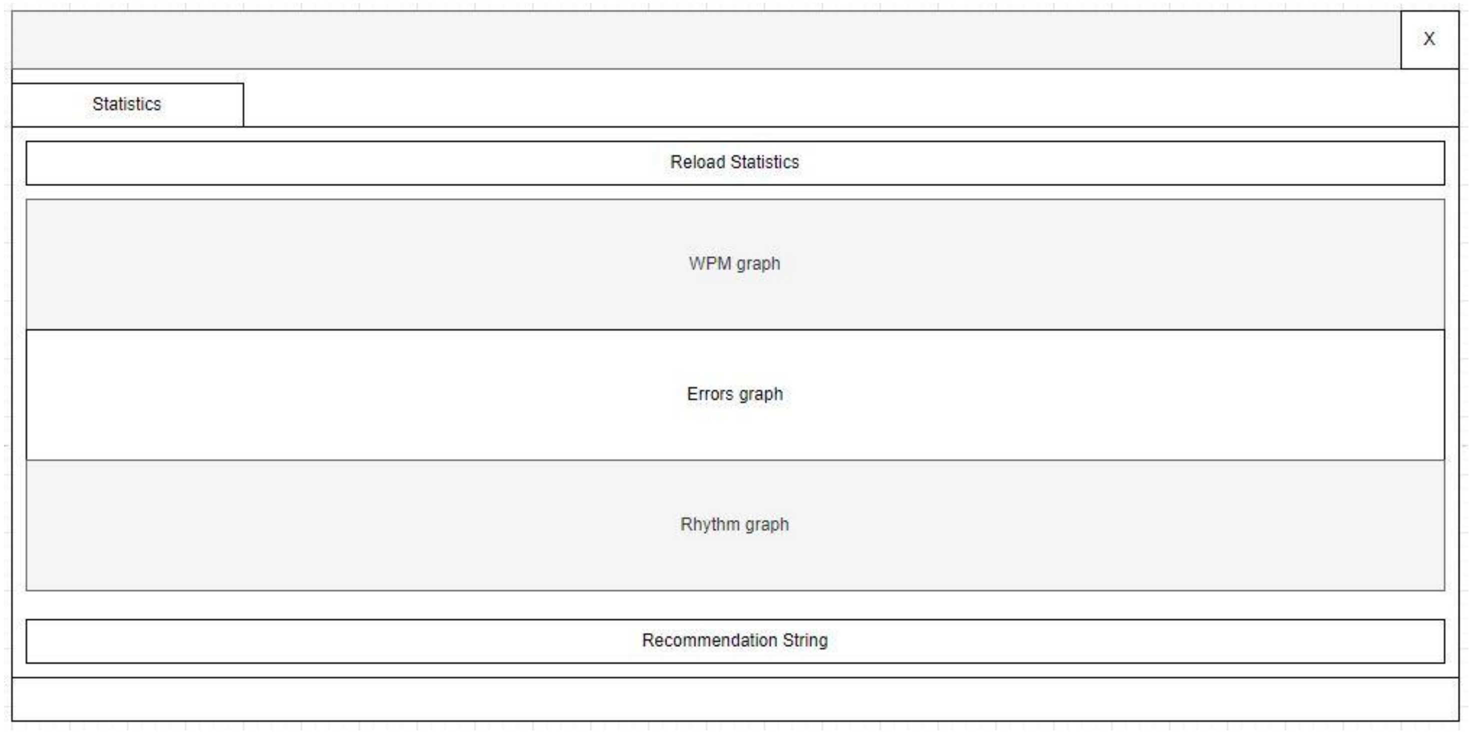
This section describes the UI of an application. Main window has two tabs:

- "Lessons and Test" where we see:
  - List of lessons with "Run Lesson" button
  - List of tests with "Run Test" button,
  - Line with a characters to type (for lesson or test)
  - Keyboard image.
- "Statistics" where user can see:
  - Three graphs of Speed, Errors and Rhythm.
  - String recommendation

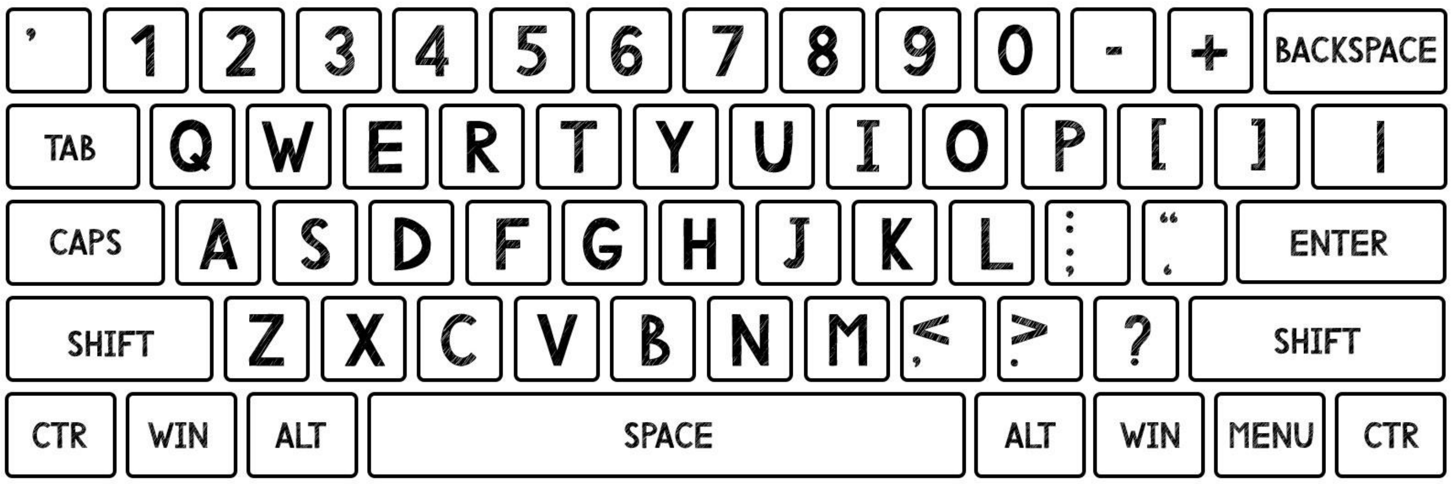
The Main window and "Lessons and Test" tab



Statistics



The Keyboard image


' 1 2 3 4 5 6 7 8 9 0 - + BACKSPACE
TAB Q W E R T Y U I O P [ ] |
CAPS A S D F G H J K L : " ENTER
SHIFT Z X C V B N M < > ? SHIFT
CTR WIN ALT SPACE ALT WIN MENU CTR

5. TESTING DESIGN

The application will be thoroughly tested at each stage of the development process, from individual components (Unit Testing) to how they interact (Integration Testing) and the overall system (System Testing). The end users will also perform User Acceptance Testing (UAT) to ensure the application meets their needs.