Typing Speed Tester

By Team **MegPie**

Abdur Rahman Sifat	ID: 22235103364
Sajidul Islam Samin	ID: 22235103396
Shahriar Choudhury	ID: 22235103389
MD.Mahamudul Hasan	ID: 22235103137
Mahdi Al Mahmud	ID: 22235103362

Submitted in partial fulfillment of the requirements of the degree of

Bachelor of Science in

Computer Science and Engineering

Department of Computer Science and Engineering

Bangladesh University of Business and Technology (BUBT)

December, 2023

Contents

Declaration

We do hereby declare that the research works presented in this thesis entitled, "Thesis/Report Title" are the results of our own works. We further declare that the thesis has been compiled and written by us and no part of this thesis has been submitted elsewhere for the requirements of any degree, award or diploma or any other purposes except for publications. The materials that are obtained from other sources are duly acknowledged in this thesis.

Signature of Authors

Abdur Rahman Sifat

ID: 22235103364

Sajidul Islam Samin

ID: 22235103396

Shahriar Chowdhury

ID:22235103389

MD.Mahamudul Hasan

ID: 22235103137

Mahdi Al Mahmud

ID:22235103362

Approval

This letter is to certify that the project entitled "Typing Speed Tester" has been successfully completed by the MagPie team, consisting of Abdur Rahman Sifat (ID: 22235103364), Sajidul Islam Samin (ID: 22235103396), MD.Mahamudul Hasan (ID: 22235103137), Shahriar Choudhury (ID: 22235103389), and Mahdi Al Mahmud (ID: 22235103362), under my supervision.

I am pleased to state that the project was completed to a high standard and meets all of the requirements for a first-year Bachelor of Science in Computer Science and Engineering (CSE) project. The students demonstrated a deep understanding of the concepts involved in the project, and their implementation was well-executed and efficient.

Supervisor

Humayra Ahmed

Assistant Professor

Department of Computer Science & Engineering

Bangladesh University of Business and Technology

Dhaka, Bangladesh

Chairman

Md.Saifur Rahman

Assistant Professor & Chairman (Acting)

Department of Computer Science & Engineering

Bangladesh University of Business and Technology

Dhaka, Bangladesh

Acknowledgement

We are deeply thankful to Bangladesh University of Business and Technology (BUBT) for providing us such a wonderful environment to peruse our research. We would like to express our sincere gratitude to Humayra Ahmed (Assistant Professor), Department of CSE, BUBT. We have completed our project with her help. We found the project area, topic, and problem with her suggestions. She guided us with our study, and supplied us many academic resources in this project. She is patient and responsible. When we had questions and needed her help, she would always find time to meet and discuss with us no matter how busy he was. We also want to give thanks to Md.Saifur Rahman, Assistant Professor & Chairman (Acting), Department of CSE, BUBT.

Chapter 1

1 Introduction

1.1 Introduction

In the fast-paced digital era, the ability to type efficiently is a valuable skill. The Typing Speed Tester project aims to provide a user-friendly platform for individuals to assess and enhance their typing speed and accuracy. This project is implemented in C++, utilizing various concepts of programming and user interface design.

1.2 Problem Statement

In the fast-paced world of digital communication, the art of typing swiftly and accurately is a valuable skill. However, many find themselves without a fun and accessible tool to measure and enhance their typing abilities. The challenge is to create a captivating Typing Speed Tester project – an interactive platform in C++ that not only evaluates users' typing speed and accuracy but also keeps them motivated through engaging exercises , receive real-time feedback, and track their progress using a scoring system over time. The goal is to transform typing practice into an enjoyable experience, making it easy for users to embark on a journey of continuous improvement.

1.3 Project Objectives

- **User-Friendly Interface:** The program incorporates an attractive and user-friendly interface, making it accessible to individuals of all skill levels.
- **Performance Evaluation:** Provide correct and insightful metrics to Users, which include WPM and accuracy, to help them understand and enhance their typing skill ability.
- **Scoring Mechanism:** A scoring mechanism is implemented to quantify the user's proficiency and motivating them to improve over time .
- **User Records:** The project maintains records of user performances, allowing individuals to track their progress and set goals for improvement.

1.4 Motivations

The Typing Speed Tester serves as a friendly coach, providing real-time feedback and tracking progress. It turns typing practice into an exciting journey of improvement, aiming to boost confidence and transform users into speedy and accurate typists. Embrace the challenge, enjoy progress, and let fingers dance on the keys with newfound skill and joy!

1.5 Project Organization

- **Chapter 2: Project Review**

  This chapter provides an overview of the Typing Speed Tester project, emphasizing its user-friendly C++ program designed to enhance typing skills. It explores the project's features, including speed tests, score storage, and a visually appealing interface. The summary hints at potential future enhancements.

- **Chapter 3: Proposed Model and Implementation**

  Chapter three outlines the feasibility, requirements, and methodology of the Typing Speed Tester project. It covers the project's technical, economic, operational, and schedule feasibility. The proposed 9-step methodology includes data collection, preprocessing, feature selection, algorithm design, and implementation details.

- **Chapter 4: Conclusion**

  The concluding chapter highlights the Typing Speed Tester as a valuable tool for skill improvement, offering an engaging way to assess typing speed and accuracy. It mentions the use of random paragraphs, metric calculations, and insights for users of all levels to track progress and enhance their typing abilities

1.6 Summary

Chapter 1 introduces the Typing Speed Tester project, highlighting its relevance in addressing the growing need for an effective tool to evaluate and enhance typing skills. The chapter provides a brief overview of the project's objectives and outlines the purpose of the report. It sets the context for the subsequent chapters. The focus is on the significance of typing proficiency in the digital era and the project's aim to fill a gap in accessible and user-friendly typing assessment platforms.

Chapter 2

2 Project Review

2.1 Project Overview

The Typing Speed Tester is like a game that helps Us get better at typing on the computer. It looks nice with different colors and easy to use. We can do speed tests to see how fast We can type and even check our past scores. Storing scores in a file is cool because we can see how much we've improved over time.Overall, it's a positive tool for improving typing skills in a fun and interactive way.

2.2 Project Analysis

The Typing Speed Tester is a console-based C++ program designed to assess a user's typing speed, accuracy, and provide a score based on their performance. The project incorporates various features and functionalities to create an interactive and engaging experience for users interested in evaluating their typing skills.

Key Features:

1. User Interaction:

- The program starts with a welcome message and prompts the user to enter their name.
- The main menu provides options for different functionalities, including speed testing, viewing user records, accessing help, and exiting the program.

2. Speed Test:

- The speed test presents the user with random sentences to type.
- After completing the typing task, the program evaluates the user's performance, calculating words per minute (WPM), accuracy percentage, and assigning a score.

3. User Records:

- Users can view their past performance records, including their name, letter per minute (LPM), and score.
- The records are stored in a file ("score.txt") for future reference.

4. Help Section:

- The help section provides information about the project, its purpose, and contact details for the development team.
- Users can access this section for additional guidance on using the Typing Speed Tester.

5. Colorful Interface:

- The program utilizes colored text to enhance the visual appeal and create a more engaging user interface.
- Different colors are used for headings, messages, and menu options.

6. Extras Section:

- The extras section displays a banner with the project name and acknowledges the development team's rights.

7. Delay and Clear Screen Functions:

- The program includes functions for introducing delays and clearing the console screen, contributing to a smoother user experience.

2.3 Summary

In Chapter 2, we delve into the exciting world of the Typing Speed Tester project. Picture this: a virtual playground for honing your typing prowers in the realm of computers. The program isn't just a run-of-the-mill typing tool. It's a vibrant game with colours that pop and a user-friendly interface that makes typing practice a joy. The Typing Speed Tester isn't just about functionality. it's a visual feast.Now, imagine future possibilities! A graphical interface could make it even more visually appealing. How about challenging friends in real-time typing battles or customizing tests to suit our fancy? The future might bring data analysis tools, offering insights into our typing trends.

Chapter 3

3 Proposed Model and Implementation

3.1 Introduction

The development of Typing Speed Tester project involves a multifaceted exploration, covering various critical aspects essential for its successful execution. This chapter embarks on a comprehensive journey through the project's Feasibility Analysis, Requirement Analysis, The Project Methodology, Data Set details, Data Collection and Preprocessing, Feature Selection, Algorithmic intricacies, and the Design and Implementation considerations.

Project Code:

```
#include <bits/stdc++.h>
#include <unistd.h>
using namespace std;
int duration, wrong = 0, score;
string name;
char uname[30];
FILE *fp;

const string textColorRed = "\x1B[31m";
const string textColorGreen = "\x1B[32m";
const string textColorReset = "\x1B[0m";
const string backgroundColorYellow = "\x1B[43m";
const string textColorBlue = "\x1B[34m";
const string textColorPurple = "\x1B[35m";

void heading();
void mainMenu();
void delay(unsigned int microseconds);
void clearScreen();
void extras();
void Speed_Test();
void calculation();
void helpSection();
```

Figure 3.1: Code

3.2 Programme Description

- The figure 3.1 shows a text-based console application that offers a series of features to users, including Speed test,Help,User record information.

```
int main()
{

    heading();
            mainMenu();
                extras();
                    getchar();
    return 0;
}
void heading()
{
    clearScreen();
    string massage = " Welcome To Typing Speed Tester ";
    cout << "\n\n\t\t\t\t\t\t";
    cout << textColorGreen;
    printf("\x1B[1m");
    printf("\x1B[4m");
    for (size_t i = 0; i < massage.length(); i++)
    {
        cout << massage[i];
        fflush(stdout);
        delay(50000);
    }
    printf("\x1B[0m");
    printf("\x1B[0m");
    cout << textColorReset;
    cout << endl
         << endl
         << endl
         << endl;

    string nameMessage = "Enter your name to continue : ";
    for (size_t i = 0; i < nameMessage.length(); i++)
    {
        cout << nameMessage[i];
        fflush(stdout);
        delay(50000);
    }

    cin >> uname;
    cout << endl;
    cout << "Hi " << uname << ","
         << " Welcome to Typing Speed Tester" << endl
         << endl;
}
```

Figure 3.2: Code

3.3 Programme Description

- The main function consist of following functions - heading(),mainMenu(),extras(). This sets the stage for the main menu, presented by the mainMenu function, where users get to choose their typing speed, view records, or access help. The extras function provides additional information, and the program waits for user input before concluding.

```
void mainMenu()
{
    clearScreen();

    int choice;

    string c_massage = "Enter Your Choice : ";
    cout << "                                                    "
         << " Menu List" << endl;
    cout << "                                                  _________________" << endl
         << endl;
    cout << textColorGreen;
    cout << "                                                   1. Test Speed" << endl;
    cout << textColorBlue;
    cout << "                                                   2. User Record\n";
    cout << textColorGreen;
    cout << "                                                   3. Help\n";
    cout << textColorRed;
    cout << "                                                   4. Exit\n\n\n";
    cout << textColorReset;
    for (size_t i = 0; i < c_massage.length(); i++)
    {
        cout << c_massage[i];
        fflush(stdout);
        delay(50000);
    }

    fflush(stdin);
    cin >> choice;
```

Figure 3.3: Code

3.4 Programme Description

- The mainMenu function presents a user-friendly menu interface, clearing the screen and displaying options for the Typing Speed Tester application. Users are prompted to enter their choice from the menu, including options to test typing speed, view user records, access help, or exit the program. The function incorporates color-coded text for clarity and utilizes a delay effect for a smoother visual presentation.

```
if (choice == 1)
{
    // extras();
    fflush(stdout);
    delay(300000);
    Speed_Test();
}
else if (choice == 2)
{
    clearScreen();
    char uname[50];
    int c_choice;
    char footer[] = {"Press 0 For Go To Main Menu : "};

    fp = fopen("score.txt", "r");
    printf("\n\nUser Name & Score\n\n");
    while (fgets(uname, 50, fp) != NULL)
    {
        cout << textColorPurple;
        printf("\t%s", uname);
    }
    cout << textColorReset;
    fclose(fp);

    printf("\n\n\t");
    for (int i = 0; i < 30; i++)
    {
        cout << textColorPurple;
        printf("%c", footer[i]);
        fflush(stdout);
        delay(20000);
    }
    cout << textColorReset;
    scanf("%d", &c_choice);

    if (c_choice == 0)
        mainMenu();
}
```

Figure 3.4: Code

3.5 Programme Description

- in this section, if the user's choice is 1, the program initiates a delay before proceeding to the Speed_Test function, likely for a visual effect or to simulate a pause. If the choice is 2, the program displays a user record section by reading data from a file ("score.txt") and presenting it on the screen. Users are prompted to return to the main menu by entering '0', triggering the mainMenu function. The use of color-coded text enhances the visual appeal of the displayed information.

```
    else if (choice == 3)
    {
        int c_choice;
        helpSection();
        cin >> c_choice;

        if (c_choice == 0)
            mainMenu();
    }
    else if (choice == 4)
    {
        clearScreen();
        string lastm = "\t\t\t\tThanks for visiting Typing Speed Tester 💜💜💜";
        cout << "\n\n\n\n\n\n\t\t\t";
        for (size_t i = 0; i < lastm.length(); i++)
        {
            cout << lastm[i];
            fflush(stdout);
            delay(50000);
        }
        cout << "\n";

        string lastn = "\t\t\t....Wait....";
        cout << "\n\n\t\t\t\t\t\t";
        for (size_t i = 0; i < lastn.length(); i++)
        {
            cout << lastn[i];
            fflush(stdout);
            delay(50000);
        }
        cout << "\n\n\n\n\n\n\n\n";
        extras();
        for (size_t i = 1; i < 1e9; i++)
        {
        }
        exit(0);
    }
}
```

Figure 3.5: Code

3.6 Programme Description

- In this code segment 3.5 corresponds to the handling of menu choice 3 and 4. If the user chooses option 3, it invokes the helpSection() function, likely displaying information or instructions. The user is prompted to enter a choice, and if it's 0, the program returns to the main menu. If the choice is 4, the program displays a farewell message, initiates a brief waiting period, and then exits the program after executing some additional functions in the extras() section. The exit is delayed for a noticeable period, providing a closing effect before terminating the program.

```
void extras()
{
    cout << endl
         << endl;
    cout << textColorRed;
    cout << ("\t\t\t\t\t\t############################################################\n");
    cout << ("\t\t\t\t\t\t##                                                        ##\n");
    cout << textColorGreen;
    cout << ("\t\t\t\t\t\t##   All rights reserved by Team 'MagPie' 2023       ##\n");
    cout << textColorRed;
    cout << ("\t\t\t\t\t\t##                                                        ##\n");
    cout << ("\t\t\t\t\t\t############################################################\n");
    cout << textColorReset;
    cout << endl
         << endl
         << endl;
    fflush(stdout);
    delay(1000000);
}
```

Figure 3.6: Code

3.7 Programmme Description

- **All headers:**
  - The figure 3.6 captures the extra function which provides the information of the Typing Speed Tester developer team follows the copyright law and all rights are reserved by team MagPie.

```
void Speed_Test()
{
    int chk[10];
    srand(time(0));
    for (int i = 0; i < 3; i++)
    {
        chk[i] = rand() % 10;
        fflush(stdout);
        delay(50000);
    }

    clearScreen();

    time_t start, end;
    start = time(NULL);
    char line1[] = {"The quick brown fox jumps over the lazy dog."};
    char line2[] = {"The quick brown fox jumps over the lazy dog."};
    char line3[] = {"tHe qUiCk bRoWn fOx jUmPs oVeR tHe lAzY dOg."};
    char line4[] = {"tHE QUICK BROWN FOX JUMPS OVER THE LAZY DOG."};
    char line5[] = {"THe qUIck BroWN fOX jUMps oVeR thE lAzY dOG."};
    char line6[] = {"tHE QUICK BROWN FOX JUMPS OVER THE LAZY DOG."};
    char line7[] = {"the Quick Brown Fox Jumps Over The Lazy Dog."};
    char line8[] = {"tHe qUiCk bRoWn fOx jUmPs oVeR tHe lAzY dOg."};
    char line9[] = {"tHe qUiCk bRoWn fOx jUmPs oVeR tHe lAzY dOg."};
    char line10[] = {"tHe qUiCk bRoWn fOx jUmPs oVeR tHe lAzY dOg."};
```

Figure 3.7: Code

3.8 Programme Description

- The Speed_Test function generates a random sequence of indices in the range [0, 9] and stores them in the array chk. It then clears the screen and initializes a timer. The function presents ten different lines of text with variations in letter casing, simulating a typing speed test. The user is likely required to replicate the provided sentences accurately within a time limit, evaluating their typing speed and accuracy. The inclusion of randomization enhances the test's variability, making it more challenging for the user.

```
for (int i = 0; i < 3; i++)
{
    if (chk[i] == 0)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line1);
    }
    else if (chk[i] == 1)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line2);
    }
    else if (chk[i] == 2)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line3);
    }
    else if (chk[i] == 3)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line4);
    }
```

```
    else if (chk[i] == 4)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line5);
    }
    else if (chk[i] == 5)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line6);
    }
    else if (chk[i] == 6)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line7);
    }
    else if (chk[i] == 7)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line8);
    }
    else if (chk[i] == 8)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line9);
    }
```

Figure 3.8: Code

```
    }
    else if (chk[i] == 9)
    {
        printf("\n\n");

        printf("\t\t\t");
        cout << textColorGreen;
        printf("%s", line10);
        fflush(stdout);
    }
}
cout << textColorReset;

// printf("\nType Here..\n\n");
cout << textColorRed;
string TypeCommand = "\nType Here..\n\n\n\n";
for (size_t i = 0; i < TypeCommand.length(); i++)
{
    cout << TypeCommand[i];
    fflush(stdout);
    delay(50000);
}
cout << textColorReset;

char testL1[32];
char testL2[32];
char testL3[32];
// cout<<backgroundColorYellow;
cout << textColorPurple;
if (chk[0] == 0)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line1[i])
            wrong++;
    }
}
```

Figure 3.9: Code

3.9 Programme Description

- This portion of the code presents three lines randomly selected from a set of predefined text lines (line1 to line10) to the user. The user is then prompted with a "Type Here" command, and their input for the first line is compared character by character. Any discrepancies are tracked, contributing to the 'wrong' counter that monitors input accuracy. Additionally, visual effects using different text colors are applied to enhance the user interface.

```
else if (chk[0] == 1)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line2[i])
            wrong++;
    }
}
else if (chk[0] == 2)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line3[i])
            wrong++;
    }
}
else if (chk[0] == 3)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line4[i])
            wrong++;
    }
}
```

```
else if (chk[0] == 4)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line5[i])
            wrong++;
    }
}
else if (chk[0] == 5)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line6[i])
            wrong++;
    }
}
else if (chk[0] == 6)
{
    fflush(stdin);
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line7[i])
            wrong++;
    }
}
else if (chk[0] == 7)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line8[i])
            wrong++;
    }
}
```

Figure 3.10: Code

```
else if (chk[0] == 8)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line9[i])
            wrong++;
    }
}
else if (chk[0] == 9)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL1[i]);
        if (testL1[i] != line10[i])
            wrong++;
    }
}

if (chk[1] == 0)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line1[i])
            wrong++;
    }
}
```

```
else if (chk[1] == 1)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line2[i])
            wrong++;
    }
}
else if (chk[1] == 2)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line3[i])
            wrong++;
    }
}
else if (chk[1] == 3)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line4[i])
            wrong++;
    }
}
else if (chk[1] == 4)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line5[i])
            wrong++;
    }
}
```

Figure 3.11: Code

```
else if (chk[1] == 5)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line6[i])
            wrong++;
    }
}
else if (chk[1] == 6)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line7[i])
            wrong++;
    }
}
else if (chk[1] == 7)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line8[i])
            wrong++;
    }
}
else if (chk[1] == 8)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL2[i]);
        if (testL2[i] != line9[i])
            wrong++;
    }
}
else if (chk[1] == 9)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
```

```
else if (chk[2] == 9)
{
    fflush(stdin);
    printf("\t\t\t");
    for (size_t i = 0; i < sizeof(line1) - 1; i++)
    {
        scanf("%c", &testL3[i]);
        if (testL3[i] != line10[i])
            wrong++;
    }
}

cout << textColorReset;
end = time(NULL);
duration = difftime(end, start);

calculation();
}
```

Figure 3.12: Code

3.10 Programme Description

- In this figure it is shown that the code takes input from the user in the form of characters. The code checks the validity of the characters based on certain conditions and updates a count of 'wrong' characters if any conditions are not met.

```
void calculation()
{
    int total_words_in_lines = 10 * 16;
    int total_letters_in_lines = 48 * 10;
    int average_letters_per_word = total_letters_in_lines / total_words_in_lines;
    double percent, right;
    int LPM, a_choice;
    char s_ch = '%';

    right = 48 - wrong;
    percent = (100.0 * right) / 48;

    if (100 >= percent && percent >= 90)
        score = 100;
    else if (90 > percent && percent >= 80)
        score = 90;
    else if (80 > percent && percent >= 70)
        score = 80;
    else if (70 > percent && percent >= 60)
        score = 70;
    else if (60 > percent && percent >= 50)
        score = 60;
    else if (50 > percent && percent >= 40)
        score = 50;
    else if (40 > percent && percent >= 30)
        score = 40;
    else if (30 > percent && percent >= 20)
        score = 30;
    else if (20 > percent && percent >= 10)
        score = 20;
    else if (10 > percent && percent >= 1)
        score = 10;
    else
        score = 0;
```

```
LPM = (48 * 60) / duration;
double WPM = LPM / average_letters_per_word;

// printing users result...
//  system("cls");
clearScreen();
char s_massage[] = {"Mr. "};
char r_massage[] = {" Here is your Result: "};

printf("\n\n\t\t");
// cout<<"👏 👏 ";
for (int i = 0; i < 4; i++)
{
    cout << textColorBlue;
    printf("%c", s_massage[i]);
    fflush(stdout);
    delay(100000);
}
cout << textColorReset;
printf("\x1B[1m");
for (size_t i = 0; i < strlen(uname); i++)
{
    cout << textColorGreen;
    printf("%c", uname[i]);
    fflush(stdout);
    delay(100000);
}
printf("\x1B[0m");
cout << textColorReset;
for (int i = 0; i < 22; i++)
{
    cout << textColorBlue;
    printf("%c", r_massage[i]);
    fflush(stdout);
    delay(100000);
}
// cout<<"👏 👏 ";
cout << textColorReset;
cout << textColorGreen;
printf("\n\tWord Per Minute is      : %.2lf", WPM);
printf("\n\n\tPercentage Of Right is  : %.2lf%c", percent, s_ch);
printf("\n\tTotal Score of Yours is : %d\n", score);
cout << textColorReset;
```

Figure 3.13: Code

3.11 Programme Description

- This function, named calculation, computes various performance metrics for the user's typing test. It calculates the user's accuracy percentage, assigns a score based on the percentage achieved, and determines the Words Per Minute (WPM) and Letters Per Minute (LPM) typing speeds. The results are then presented in a visually appealing format, including a personalized message addressing the user by name and displaying the achieved WPM, accuracy percentage, and overall score. Visual effects using different text colors are employed for an enhanced user interface.

```
    fp = fopen("score.txt", "a");
    fprintf(fp, "Name: %s\tWord Per Minutes: %.2lf\tScore: %d\n", uname, WPM, score);
    fclose(fp);

    string footer1 = "1. Test Again  ";
    string footer2 = "2. Return Menu List ";
    string footer3 = "Enter Your Choice :";
    printf("\n\n\t");
    cout << textColorGreen;
    for (size_t i = 0; i < footer1.length(); i++)
    {
        printf("%c", footer1[i]);
        fflush(stdout);
        delay(20000);
    }
    printf("\n\t");
    cout << textColorReset;
    cout << textColorRed;
    for (size_t i = 0; i < footer2.length(); i++)
    {
        printf("%c", footer2[i]);
        fflush(stdout);
        delay(20000);
    }
    printf("\n\n\t");
    cout << textColorReset;
    cout << textColorPurple;
    for (size_t i = 0; i < footer3.length(); i++)
    {
        printf("%c", footer3[i]);
        fflush(stdout);
        delay(20000);
    }
    cout << textColorReset;
    cout << textColorPurple;
    scanf("%d", &a_choice);
    if (a_choice == 1)
    {
        wrong = 0;
        Speed_Test();
    }

    else if (a_choice == 2)
        mainMenu();
    cout << textColorReset;
}
```

Figure 3.14: Code

3.12 Programme Description

- This code snippet appends the user's performance details, including name, Words Per Minute (WPM), and score, to a file named "score.txt". It then displays a menu with options to either take the test again or return to the main menu. The user's choice is captured, and if they choose to test again, the typing test function (Speed_Test()) is called with the wrong count reset. If they choose to return to the main menu, the main menu function (mainMenu()) is invoked. The user interface is enhanced with different text colors for a more engaging experience.

```
void delay(unsigned int microseconds)
{
    usleep(microseconds);
}
void clearScreen()
{
    printf("\033[H\033[J");
}
void helpSection()
{
    int c_choice;
    string footer = "Press 0 For Go To Main Menu : ";
    string HelpMessage = "\n\n * If you want to test your typing speed, this is the best place for you.\n\n
    * If you want to test at first you need to input the value of 1, 1 is the function which takes you to the test page \n\n
      where you can type some letter and test your LPM(Letter Per Minutes), and accuracy as well as score \n\n
      * If you want more information you can contact us: \n\n\n\t\t\t\t\tEmail:  \n\n\t\t\t\tsupport.magpie@hotmail.co\n\n\t\t\t\t";

    for (size_t i = 0; i < HelpMessage.length(); i++)
    {
        cout << textColorPurple;
        cout << HelpMessage[i];
        fflush(stdout);
        delay(20000);
    }
    cout << textColorReset;
    printf("\n\n\t");
    for (size_t i = 0; i < footer.length(); i++)
    {
        cout << textColorGreen;
        printf("%c", footer[i]);
        fflush(stdout);
        delay(20000);
    }
    cout << textColorReset;
    scanf("%d", &c_choice);

    if (c_choice == 0)
        mainMenu();
}
```

Figure 3.15: Code

3.13 Programme Description

- This code defines three functions: delay, clearScreen, and helpSection. The delay function introduces a pause in the program for a specified duration using the usleep function. The clearScreen function is responsible for clearing the console screen. The helpSection function provides information about the typing speed testing program, including instructions on how to test typing speed. It displays a help message with contact information and prompts the user to press 0 to return to the main menu. Upon user input, it either returns to the main menu or continues with the selected option. The text is displayed with colorful formatting for an enhanced user interface.

3.14 Project Output:

```
                                    Welcome To Typing Speed Tester


Enter your name to continue : MagPie

Hi MagPie, Welcome to Typing Speed Tester

                                                        Menu List
                                                    ________________

                                                     1. Test Speed
                                                     2. User Record
                                                     3. Help
                                                     4. Exit

Enter Your Choice : 1

                        The quick brown fox jumps over the lazy dog.

                        tHE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.

                        tHe qUiCk bRoWn fOx jUmPs oVeR tHe lAzY dOg.
Type Here..


                        tHE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
                        tHE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
                        tHe qUiCk bRoWn fOx jUmPs oVeR tHe laxy god.
```

Figure 3.16: Speed Test Interface

```
        Mr. MagPie Here is your Result:
Word Per Minute is      : 6.00

Percentage Of Right is  : 14.58%
Total Score of Yours is : 20


1. Test Again
2. Return Menu List

Enter Your Choice :
```

```
User Name & Score

        Name: sifat     Letter Per Minutes: 169 Score: 100
        Name: sifat     Letter Per Minutes: 56  Score: 100
        Name: sfhgjfvdxgsdgk    Letter Per Minutes: 192 Scor    e: 100
        Name: sdlhjcvadsiu      Letter Per Minutes: 120 Score:   100
        Name: sdlhjcvadsiu      Letter Per Minutes: 151 Score:   100
        Name: s Word Per Minutes: 50.00 Score: 80
        Name: sifat     Word Per Minutes: 50.00 Score: 80
        Name: s Word Per Minutes: 43.00 Score: 100
        Name: sifat     Word Per Minutes: 21.00 Score: 100
        Name: MagPie    Word Per Minutes: 6.00  Score: 20


        Press 0 For Go To Main Menu :
```

Figure 3.17: The Test Result

```
Enter Your Choice : 3

 * If you want to test your typing speed, this is the best place for you.

 * If you want to test at first you need to input the value of 1, 1 is the function which takes you to the test page

 where you can type some letter and test your LPM(Letter Per Minutes), and accuracy as well as score

 * If you want more information you can contact us:

                                    Email:

                              support.magpie@hotmail.co


       Press 0 For Go To Main Menu :
```

Figure 3.18: Users Previous Records

```
                    Thanks for visiting Typing Speed Tester 💜💜💜

                            ....Wait....



##################################################
##                                              ##
##   All rights reserved by Team 'MagPie' 2023  ##
##                                              ##
##################################################
```

Figure 3.19: Greetings

3.15 Feasibility analysis

- **Technical Feasibility:** The technical feasibility of a typing speed tester project involves evaluating technology and infrastructure availability. This includes assessing compatibility with different devices and platforms, scalability, performance requirements, reliable internet connectivity, and server infrastructure. Security measures, such as data encryption and protection against hacking, are crucial. Overall, careful consideration of the technology stack, infrastructure, and security measures ensures a smooth and reliable user experience.

- **Economic Feasibility:** The Economic Feasibility of the Typing Speed Tester project underscores its cost-effectiveness and financial sustainability. Leveraging open-source tools minimizes software acquisition costs, while the project's efficient design reduces hardware requirements. Scalability ensures optimal performance under varying user loads without substantial infrastructure investment. With low maintenance needs, the project promises long-term cost savings. In summary, the Typing Speed Test project presents an economically feasible solution, providing value with minimal upfront, operational, and maintenance expenses.

- **Operational Feasibility:** The Operational Feasibility of the Typing Speed Tester project is characterized by its user-friendly interface, minimal training requirements, and efficient data management. The system seamlessly integrates into user workflows, ensuring ease of adoption and smooth administration. With scalability to accommodate diverse user levels, the project is operationally robust, aligning effectively with user needs and streamlining administrative processes. Overall, its operational feasibility is a key strength, contributing to the project's success and user satisfaction.

- **Schedule Feasibility:** The Schedule Feasibility of the Typing Speed Tester project demonstrates a well-structured and realistic timeline for development, testing, and implementation. The project adheres to predefined milestones, allowing for efficient task management and timely delivery. Regular progress assessments and flexible adjustments ensure adaptability to unforeseen challenges, enhancing the project's overall schedule feasibility. The project's timeline aligns with the outlined goals, reflecting a commitment to on-time completion and successful integration into the designated timeframe.

3.16 Requirement Analysis

Typing Speed Tester project involves a thorough examination of functional and non-functional specifications. Functionally, the system must accurately measure typing speed and assess accuracy based on provided paragraphs. User authentication and result storage are essential. Non-functional requirements include a user-friendly interface, cross-platform compatibility, and efficient processing to ensure real-time feedback. Accessibility features for differently-abled users should also be considered. Additionally, scalability to accommodate future enhancements and security measures to protect user data are integral aspects of the project's requirement analysis. The analysis forms the foundation for the successful development and implementation of the Typing Speed Test project.

3.17 The Project Methodology

The project methodology for the Typing Speed Tester involves a systematic approach to ensure effective development and implementation. The methodology can be outlined as follows:

1. Project Planning:Define project scope, objectives, and deliverables. Develop a detailed project plan outlining tasks, timelines, and resource allocation.

2. Requirement Analysis: Conduct a thorough analysis of functional and non-functional requirements. Identify key features, user interactions, and system constraints.

3. Design:Create a detailed system design, including the user interface, backend logic, and database structure. Ensure scalability, responsiveness, and a user-friendly experience.

4. Implementation: Code the application using C++ programming language. Implement features such as user input processing, speed calculation, accuracy assessment, and result display.

5. Testing: Conduct extensive testing, including unit testing, integration testing, and user acceptance testing. Ensure the system functions accurately and meets all specified requirements.

6. Documentation: Create comprehensive documentation, including user manuals and technical documentation. This aids in system maintenance and user guidance.

7. Deployment: Deploy the Typing Speed Test application on suitable platforms. Ensure compatibility and performance in different environments.

8. User Training: Provide training materials or sessions for users to understand how to use the Typing Speed Test effectively.

9. Feedback and Iteration: Collect user feedback and make iterative improvements to enhance the application's performance, usability, and features. By following this methodology, the Typing Speed Test project can be systematically developed, ensuring a robust and user-friendly application.

3.18 Data Collection & Prepossessing

Data Collection

1. User Input: Capture the user's input as they type the provided text. This includes recording the characters typed, the time taken to type each character, and any errors made.

2. Device Information: Collect information about the user's device, such as their keyboard type, browser version, and operating system. This information can be used to contextualize the typing performance data.

3. User Demographics: Optionally, collect demographic information about the user, such as their age, location, and occupation. This information can be used to analyze typing speed trends across different demographics.

Data Processing

1. Cleaning and Preprocessing: Clean the collected data to remove any inconsistencies or errors. This may involve handling typos, removing illegal characters, and normalizing timestamps.

2. Feature Extraction: Extract relevant features from the data. These features may include: - Typing Speed: Calculate the average typing speed in words per minute (WPM) and characters per minute (CPM). - Accuracy: Calculate the percentage of characters typed correctly. - Error Analysis: Identify common errors made by the user, such as typos, missed characters, and extra characters.

3. Performance Analysis: Analyze the typing speed and accuracy data to identify patterns and trends. This may involve comparing performance across different text types, user demographics, and device types.

4. Data Visualization:Create visualizations to represent the analyzed data, such as charts, graphs, and tables. These visualizations can be used to communicate insights to users and stakeholders.

5. Data Storage:Store the collected and processed data in a secure and accessible manner. This may involve using a database or a cloud storage service.

3.19 Algorithm

The Speed Typing Tester project involves several algorithms to achieve its functionalities. Here is an overview of the key algorithms implemented in the code:

1. **Typing Speed Calculation Algorithm:** Randomly selects three paragraphs for the user to type. - Measures the time taken by the user to complete typing. - Calculates the typing speed in Letters Per Minute (LPM) and Word Per Minute (WPM). - Evaluates the accuracy of the user's input.

2. **Scoring Algorithm:** Computes the percentage of correctly typed characters. Assigns a score based on the percentage achieved. Utilizes a grading system to categorize users into different proficiency levels.

3. **File Handling Algorithm:** Manages user records by reading and writing to a file . Appends new records for each typing test, including the user's name, LPM, and score.

4. **User Interface Animation Algorithm:** Utilizes ANSI escape codes for colorful and dynamic console output. Delays output to create a typewriter-like effect, enhancing user engagement. Clears the console screen for a clean and organized interface.

5. **Menu Navigation Algorithm:** Implements a menu-driven interface for user interaction. Allows users to choose between typing tests, viewing records, accessing help, and exiting the program. Facilitates navigation between different sections of the program.

6. **Help Section Algorithm:** Displays information about the purpose and usage of the typing speed tester. Provides contact details for support.

7. **Delay and Clear Screen Algorithm:** Implements delay functions to control the speed of text output, creating a smooth and visually appealing display. Clears the console screen for better readability and an organized interface.

3.20 Design and Implementation

The Speed Typing Tester project reflect a structured and user-friendly approach. The code is organized into modular functions for distinct functionalities. The project follows a console-based interface, engaging users with colorful and dynamic elements. The implementation employs features like file handling for user records and time functions for speed calculations. The code structure is well-commented, promoting readability and easy maintenance. The design incorporates a welcoming interface, and the code adheres to good programming practices, enhancing its effectiveness as a typing speed assessment tool. Overall, the project seamlessly integrates functionality with a visually appealing design for an interactive user experience.

3.21 Results and Discussion

The Typing Speed Tester project achieves accurate assessments and fosters user improvement through effective algorithms. Its user-friendly interface and scalability enhance the experience, while a comprehensive dataset enables diverse evaluations. Discussions in the Results and Discussion section underscore the project's potential to enhance typing skills, making it a valuable tool for skill development.

Chapter 4

4 Conclusion

4.1 Introduction

The Typing Speed Tester is a project that allows users to test their typing speed and accuracy. It takes a paragraph as input and calculates the user's typing speed in terms of letters per minute (LPM) and their accuracy. The project provides a user-friendly interface and displays the results in an easy-to-understand format.

4.2 Conclusion

The Typing Speed Tester project is a useful tool for individuals who want to improve their typing skills. It provides an interactive and engaging way to test typing speed and accuracy. By using random paragraphs and calculating various metrics, the project offers valuable insights into the user's typing performance. Whether you are a beginner or an experienced typist, the Typing Speed Tester can help you track your progress and enhance your typing abilities.