A

Project Report On

# Typing Master

Submitted in partial fulfillment of the requirement for the IVth semester

**Bachelor of technology (CSE)**

By

**Rahul Singh 20011781**

**Piyush Upadhyay 20011399**

**Narendra Singh Jeena 20011464**

**Under the Guidance of**

**Dr. Sashi Kumar Sharma**

**Department of Computer science and Engineering**



**SCHOOL OF COMPUTING**

**GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS**

**SATTAL ROAD, P.O. BHOWALI, DISTRICT- NAINITAL-263132**

**2021-2022**

## STUDENT’S DECLARATION

We, **Rahul SIngh , piyush upadhyay , Narendra singh Jeena** hereby declare the work, which is being presented in the project, entitled “ Typing Master ” in partial fulfillment of the requirement for the award of the degree **(B.Tech)** in the session **2021-2022**, is an authentic record of my work carried out under the supervision of Dr. Sashi kumar Sharma.

The matter embodied in this project has not been submitted by me for the award of any other degree.

Rahul singh

Date:16-06-2022 Piyush Upadhyay

Narendra singh Jeena

## 

## CERTIFICATE

**The project report entitled “Typing Master ” being submitted by Rahul Singh S/o Mr. Prem singh, 20011781 of B.Tech, Piyush Upadhyay S/o Basant upadhyay, 20011399, Nrendra singh Jeena S/o Mr. Kheem singh jeena , 20011464 to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.**

Dr. Shashi Kumar Sharma Dr. Shashi Kumar Sharma Dr. Ankur Bisht

**(Project Guide) (Class Coordinator) (HOD, CSE Department)**

## ACKNOWLEDGEMENT

We **Rahul singh, Piyush upadhyay, Narendra singh jeena** take immense pleasure in thanking the Honorable Director **“Prof. (Dr.) Manoj Chandra Lohani”,** GEHU Bhimtal Campus to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that I need. I again want to extend thanks to our chairman **“Prof. (Dr.) Kamal Ghanshala”** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to **“Dr. Ankur Bisht”** (HOD-CS , GEHU Bhimtal Campus)**, “Dr. Sashi Kumar Sharma”** , our project guide <<Guide Name and Designation>> and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this thesis.

Finally, yet importantly, I would like to express my heartiest thanks to my beloved parents, for their moral support, affection, and blessings. I would also like to pay my sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this research.

Rahul singh, 20011781

Piyush upadhyay, 20011399

Narendra singh jeena, 20011464

**TABLE OF CONTENTS**

Declaration 2

[Certificate 3](#_TOC_250002)

[Acknowledgement 4](#_TOC_250001)

[Table of Contents 5](#_TOC_250000)

1. Introduction 6
2. Proposed System 7
3. S/W And H/W Requirements 8
4. Database Design 9
5. BASIC CONCEPTS & TOOLS 10
6. Coding of Function 11
7. Outputs 12
8. Reference 13

# 

# INTRODUCTION

# 

Touch-typing, or keyboarding, isn't the most fun skill to learn, but it's one of the most important. Quality typing programs boast student-friendly interfaces, lots of customization options, powerful teacher dashboards, game-based learning, and even connections to other content areas like digital citizenship. They help struggling students with reinforcement and review and provide high interest texts for more accurate typists. These are the best typing games and comprehensive programs we've found for helping kids learn to type, use proper form, increase their typing speed, and develop good keyboarding habits.

A typing test is designed to find how fast one types in a given amount of time. We will be designing a typing game using JavaScript that presents a simple typing challenge and finds the performance of typing by calculating the Characters Per Minute (CPM), Words Per Minute (WPM) and the accuracy of the typed characters.

The game shows a series of quotes that have to be typed in a specified time limit, as fast as possible. A higher typing speed would show a higher WPM value. Incorrectly typed characters would be marked accordingly during typing. We will create the HTML layout first, style it using CSS and then write the logic in JavaScript.

**Software Requirements**

Tools Used:

• Python IDLE.

• Interpreters for scripts.

• Selenium Web driver in python.

• Google Speech-to-text and text-to-speech Converters.

**Hardware Requirements**

• Windows Desktop

HTML LAYOUT

The HTML layout defines the element structure that would be shown on the page. This includes:

* **Header Portion:** This section shows the statistics of the current typing session. This includes the display of the time left, number of errors, accuracy, WPM and CPM.
* **Quote Section:** This section shows the current text that has to be typed in the input area.
* **Input Area:** This section contains the input area where the text has to be typed.
* **Restart Button:** This is the restart button which would be shown once the time runs out and the game finishes.

**Code:**

<html lang="en">

<head>

<title>Simple Speed Typer</title>

<!-- link the CSS file here -->

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="container">

<div class="heading">

Simple Speed Typing

</div>

<div class="header">

<div class="wpm">

<div class="header\_text">WPM</div>

<div class="curr\_wpm">100</div>

</div>

<div class="cpm">

<div class="header\_text">CPM</div>

<div class="curr\_cpm">100</div>

</div>

<div class="errors">

<div class="header\_text">Errors</div>

<div class="curr\_errors">0</div>

</div>

<div class="timer">

<div class="header\_text">Time</div>

<div class="curr\_time">60s</div>

</div>

<div class="accuracy">

<div class="header\_text">% Accuracy</div>

<div class="curr\_accuracy">100</div>

</div>

</div>

<div class="quote">

Click on the area below to start the game.

</div>

<textarea class="input\_area"

placeholder="start typing here..."

oninput="processCurrentText()"

onfocus="startGame()">

</textarea>

<button class="restart\_btn"

onclick="resetValues()">

Restart

</button>

</div>

<!-- link the JavaScript file here -->

<script src="game.js">

</script>

</body>

</html>

THE CSS STYLING

CSS is used to style the different portions and make it more visually appealing.

* The header portion is displayed using the flex layout.
* Adequate padding and margin are given to each element.
* The text size of each element is such that it is easily readable by the user when playing the game.
* Two additional classes are defined to denote the letters that are typed correctly or incorrectly. These classes would be dynamically added or removed when required.

**Code:**

body {

background-color: #fe9801;

color: black;

text-align: center;

}

.container {

display: flex;

flex-direction: column;

align-items: center;

}

.heading {

margin-bottom: 20px;

font-size: 3rem;

color: black;

}

.header {

display: flex;

align-items: center;

}

.timer, .errors, .accuracy,

.cpm, .wpm {

background-color: #ccda46;

height: 60px;

width: 70px;

margin: 8px;

padding: 12px;

border-radius: 20%;

box-shadow: black 5px 8px 5px;

}

.cpm, .wpm {

display: none;

}

.header\_text {

text-transform: uppercase;

font-size: 0.6rem;

font-weight: 600;

}

.curr\_time, .curr\_errors,

.curr\_accuracy, .curr\_cpm,

.curr\_wpm {

font-size: 2.75rem;

}

.quote {

background-color: #ccda46;

font-size: 1.5rem;

margin: 10px;

padding: 25px;

box-shadow: black 5px 8px 5px;

}

.input\_area {

background-color: #f5f5c6;

height: 80px;

width: 40%;

font-size: 1.5rem;

font-weight: 600;

margin: 15px;

padding: 20px;

border: 0px;

box-shadow: black 5px 8px 5px;

}

.restart\_btn {

display: none;

background-color: #326765;

font-size: 1.5rem;

padding: 10px;

border: 0px;

box-shadow: black 5px 8px 5px;

}

.incorrect\_char {

color: red;

text-decoration: underline;

}

.correct\_char {

color: darkgreen;

}

The result of the HTML layout and CSS styling would look like this :

Graphical user interface, application, PowerPoint

Description automatically generated

**Main Logic of the game:**

The main logic of the game is defined in a JavaScript file. There are several functions that work together to run the game.

**Step 1:** Selecting all the elements and defining variables

The required elements in the HTML layout are first selected using the querySelector() method. They are assigned variable names so that they could be easily accessed and modified. Other variables that would be accessed throughout the program are also defined in the beginning.

// define the time limit

let TIME\_LIMIT = 60;

// define quotes to be used

let quotes\_array = [

"Push yourself, because no one else is going to do it for you.",

"Failure is the condiment that gives success its flavor.",

"Wake up with determination. Go to bed with satisfaction.",

"It's going to be hard, but hard does not mean impossible.",

"Learning never exhausts the mind.",

"The only way to do great work is to love what you do."

];

// selecting required elements

let timer\_text = document.querySelector(".curr\_time");

let accuracy\_text = document.querySelector(".curr\_accuracy");

let error\_text = document.querySelector(".curr\_errors");

let cpm\_text = document.querySelector(".curr\_cpm");

let wpm\_text = document.querySelector(".curr\_wpm");

let quote\_text = document.querySelector(".quote");

let input\_area = document.querySelector(".input\_area");

let restart\_btn = document.querySelector(".restart\_btn");

let cpm\_group = document.querySelector(".cpm");

let wpm\_group = document.querySelector(".wpm");

let error\_group = document.querySelector(".errors");

let accuracy\_group = document.querySelector(".accuracy");

let timeLeft = TIME\_LIMIT;

let timeElapsed = 0;

let total\_errors = 0;

let errors = 0;

let accuracy = 0;

let characterTyped = 0;

let current\_quote = "";

let quoteNo = 0;

let timer = null;

**Step 2:** Preparing the text to be displayed

A function updateQuote() is defined which handles the following things: 

* **Getting the text**

Quotes have been used as the text that has to be typed to play the game. Each quote is taken one by one from a predefined array. A variable keeps track of the current quote index and increments it whenever a new one is requested.

* **Splitting the characters into elements**

Each of the characters in the text is separated into a series of <span> elements. This makes it possible to individually change the color of each character depending upon if it has been correctly typed by the user. These elements are appended to a variable quote\_text.

function updateQuote() {

quote\_text.textContent = null;

current\_quote = quotes\_array[quoteNo];

// separate each character and make an element

// out of each of them to individually style them

current\_quote.split('').forEach(char => {

const charSpan = document.createElement('span')

charSpan.innerText = char

quote\_text.appendChild(charSpan)

})

// roll over to the first quote

if (quoteNo < quotes\_array.length - 1)

quoteNo++;

else

quoteNo = 0;

}

**Step 3:** Getting the currently typed text by the user

A function processCurrentText() is defined which will be invoked whenever the user types or changes anything in the input box. It is hence used with the oninput event handler of the input box. This function handles the following things:

* **Getting the current value of the input box**

The value property of the input area is used to get the current text typed by the user. This is split into an array of characters to compare with the quote text. This is stored in curr\_input\_array.

* **Coloring the characters of the quote text**

The characters of the displayed quote are colored ‘red’ or ‘green’ depending on whether it has been typed correctly. This is done by selecting the span elements of the quote we have created earlier and looping through them. The element has then applied the classes created above depending on if it matches the typed text.

* **Calculating the errors and accuracy**

Every time the user makes a mistake during typing, the errors variable is incremented. This is used to calculate the accuracy value by dividing the number of correctly typed characters with the total number of characters typed by the user. 

* **Moving to next quote**

When the length of the input text matches the quote text length, the updateQuote() function is called which changes the quote and clears the input area. The number of total errors is also updated to be used for the next quote.

function processCurrentText() {

// get current input text and split it

curr\_input = input\_area.value;

curr\_input\_array = curr\_input.split('');

// increment total characters typed

characterTyped++;

errors = 0;

quoteSpanArray = quote\_text.querySelectorAll('span');

quoteSpanArray.forEach((char, index) => {

let typedChar = curr\_input\_array[index]

// character not currently typed

if (typedChar == null) {

char.classList.remove('correct\_char');

char.classList.remove('incorrect\_char');

// correct character

} else if (typedChar === char.innerText) {

char.classList.add('correct\_char');

char.classList.remove('incorrect\_char');

// incorrect character

} else {

char.classList.add('incorrect\_char');

char.classList.remove('correct\_char');

// increment number of errors

errors++;

}

});

// display the number of errors

error\_text.textContent = total\_errors + errors;

// update accuracy text

let correctCharacters = (characterTyped - (total\_errors + errors));

let accuracyVal = ((correctCharacters / characterTyped) \* 100);

accuracy\_text.textContent = Math.round(accuracyVal);

// if current text is completely typed

// irrespective of errors

if (curr\_input.length == current\_quote.length) {

updateQuote();

// update total errors

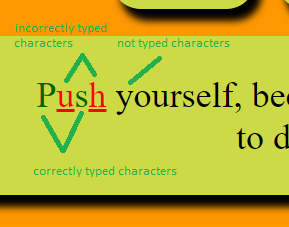
total\_errors += errors;

// clear the input area

input\_area.value = "";

}

}



**Step 4:**  Starting a new game

A function startGame() is defined which will be invoked when the user focuses on the input box. It is hence used with the onfocus event handler of the input box. This function handles the following things:

* **Reset all values**

All the values are reset to their default ones before the starting of a new game. We create a different function named resetValues() which handles this. 

* **Update the quote text**

A new quote text is made ready and displayed by calling the updateQuote () function. 

* **Creating a new timer**

A timer keeps track of the number of seconds left and displays it to the user. It is created using the setInterval() method which repeatedly calls the updateTimer() function defined below. Before creating a new timer, the previous timer instance is cleared using clearInterval().

function startGame() {

resetValues();

updateQuote();

// clear old and start a new timer

clearInterval(timer);

timer = setInterval(updateTimer, 1000);

}

function resetValues() {

timeLeft = TIME\_LIMIT;

timeElapsed = 0;

errors = 0;

total\_errors = 0;

accuracy = 0;

characterTyped = 0;

quoteNo = 0;

input\_area.disabled = false;

input\_area.value = "";

quote\_text.textContent = 'Click on the area below to start the game.';

accuracy\_text.textContent = 100;

timer\_text.textContent = timeLeft + 's';

error\_text.textContent = 0;

restart\_btn.style.display = "none";

cpm\_group.style.display = "none";

wpm\_group.style.display = "none";

}

**Step 5:**  Updating the timer

A function updateTimer() is defined which will be invoked every second to keep track fo the time. This function handles the following things:

* **Update the time values**

All the variables that keep track of the time are updated. The timeLeft value is decremented, the timeElapsed value is incremented, and the timer text is updated to the current time left.

* **Finishing the game**

This portion is triggered when the time limit is reached. It calls the finishGame () function defined below which finishes the game.

function updateTimer() {

if (timeLeft > 0) {

// decrease the current time left

timeLeft--;

// increase the time elapsed

timeElapsed++;

// update the timer text

timer\_text.textContent = timeLeft + "s";

}

else {

// finish the game

finishGame();

}

}

**Step 6:**  Finishing the game

A function finishGame() is defined which will be invoked when the game has to be finished. This function handles the following things:

* **Deleting the timer**

The timer instance created before is removed.

* **Displaying the restart game text and button**

The quoted text displayed to the user is changed to one that indicates that the game is over. The ‘Restart’ button is also displayed by setting the display property to ‘block’.

* **Calculating the CPM and WPM of the current session**
  1. The Characters Per Minute (CPM) is calculated by dividing the total number of characters typed with the time elapsed and then multiplying the result with 60. It is rounded off to prevent decimal points.
  2. The Words Per Minute (WPM) is calculated by dividing the CPM by 5 and then multiplying the result with 60. The 5 denotes the average number of characters per word. It is rounded off to prevent decimal points.

function finishGame() {

// stop the timer

clearInterval(timer);

// disable the input area

input\_area.disabled = true;

// show finishing text

quote\_text.textContent = "Click on restart to start a new game.";

// display restart button

restart\_btn.style.display = "block";

// calculate cpm and wpm

cpm = Math.round(((characterTyped / timeElapsed) \* 60));

wpm = Math.round((((characterTyped / 5) / timeElapsed) \* 60));

// update cpm and wpm text

cpm\_text.textContent = cpm;

wpm\_text.textContent = wpm;

// display the cpm and wpm

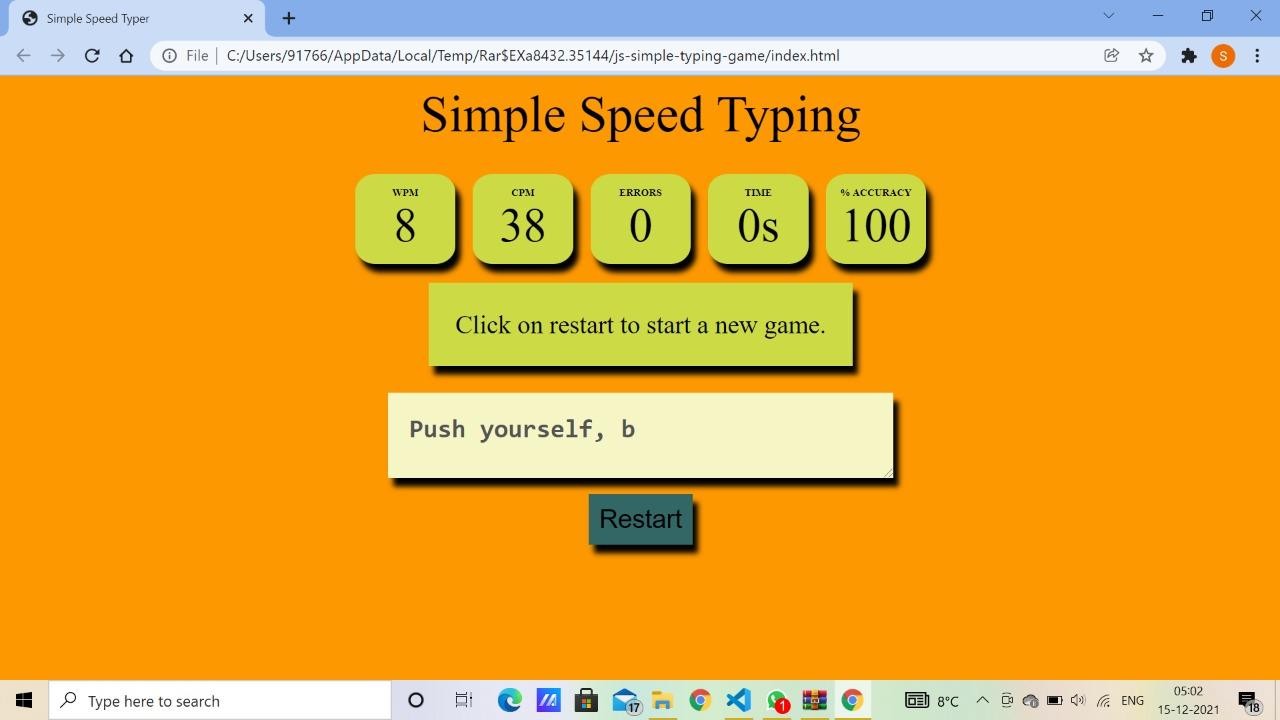
cpm\_group.style.display = "block";

wpm\_group.style.display = "block";

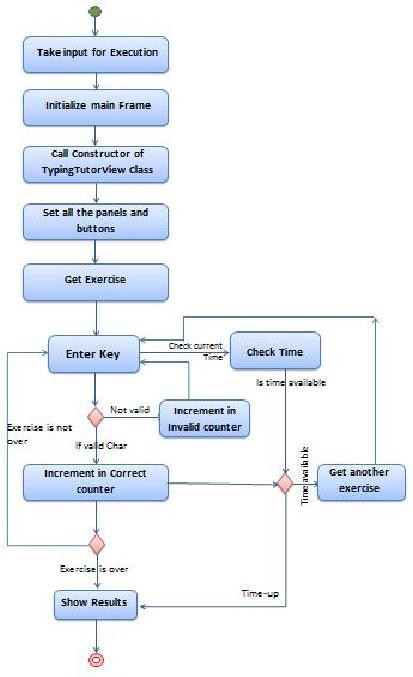
}

**Final Demonstration**

The game is now ready to be played in any browser.



Flow chart



**References**

1. w3School
2. Stack overflow
3. Geeks for geeks