**SINAI UNIVERSITY FACULTY OF**
**INFORMATION TECHNOLOGY AND**
**COMPUTER SCIENCE**

# *Typing Speed Test Project Documentation*

**by**



| Student Name | Student ID |
|---|---|
| Mostafa Fouad Ali | 202201252 |
| Saleh Ali Mahmoud | 202202873 |
| Ahmed Khalaf Abdelghani | 202202611 |



**Submitted in partial fulfillment of the degree for the requirements for the degree of B.Sc. in Information Technology.**

## Under the Supervision of

**Prof :Dr. Nasser Tamim**

Department of Information Technology

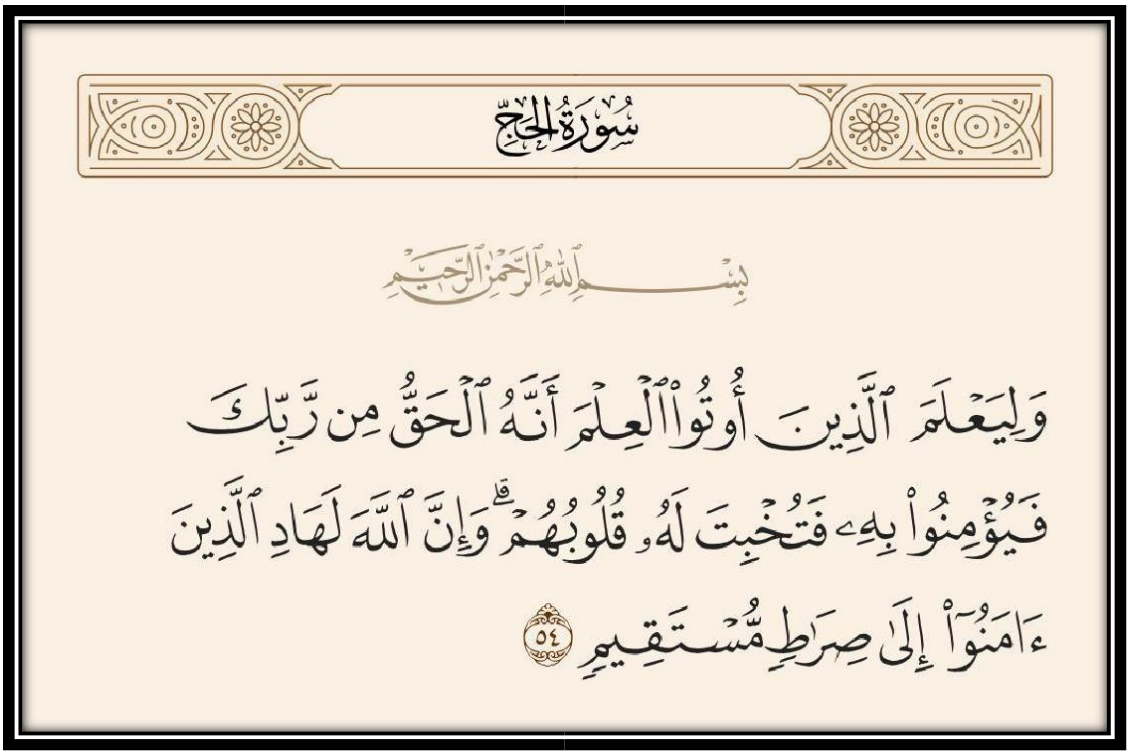
We clarify that we have read this graduation project report as examining committee, examined the students in its content and that in our opinion it is adequate as a project document for B.Sc. in Information Technology.

| Supervisor | Examiner One | Examiner Two |
|---|---|---|
| **Dr.Nasser Tamim** | | |
| …… | …… | …… |

# D E C L A R A T I O N

We hereby certify that this material, which I now submit for assessment on the program of study leading to the award of Bachelors of Science in Information Technology, is entirely our own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

| | |
|---|---|
| Mostafa Fouad Ali | **Signed: ……..** |
| Saleh Ali Mahmoud | **Signed: ……..** |
| Ahmed Khlaf Abdelghani | **Signed: ……..** |




**صدق الله العظيم.**

# A C K N O W L E D G M E N T

We would like to give our greatest appreciation to our project advisor, ***Dr. Nasser Tamim,*** Without his guidance, this project would never have been completed so well. He continuously provided us great ideas to improve both our program and our programming ability. Also, we want to thank ***Faculty of Information Technology and Computer Science, Sinai University*** for giving us this chance to get into the field of *information technology*. All the staff and faculty always try their best to assist us whenever we needed them to. They also worked as hard as possible to make our learning environment better every day. Without them, we would never have this opportunity to finish our first software project as programmers.

# Typing Speed Test Project Documentation

## A Comprehensive Guide:

## Table of Contents

### 1. Introduction

- **1.1 Project Overview**
  This project aims to develop a web-based typing speed test application. The application will provide users with a platform to assess their typing proficiency by measuring their words per minute (WPM) and accuracy. It will offer various text samples and difficulty levels, along with a user-friendly interface to track and display results.

- **1.2 Purpose and Goals**
  The primary purpose of this project is to create a reliable and accessible tool for individuals to improve their typing skills. The goals are:
  - To accurately measure typing speed and accuracy.
  - To provide a user-friendly and engaging experience.
  - To offer customizable test parameters (e.g., text length, difficulty).
  - To store and display user results for progress tracking.
  - To be accessible via common web browsers.

- **1.3 Target Audience**
  The target audience includes:
  - Individuals seeking to improve their typing skills.
  - Students and professionals who require efficient typing.
  - Individuals preparing for typing proficiency tests.
  - Anyone interested in assessing their typing speed.

## 2. Project Planning

- **2.1 Scope**
  The scope of this project includes the development of a web application that:
  - Generates random text samples for typing tests.
  - Calculates WPM and accuracy in real-time.
  - Displays test results and historical data.
  - Provides adjustable test settings.

- **2.2 Features**
  - Real-time WPM and accuracy calculation.
  - Display of typed text and errors.
  - Timer for test duration.
  - Selection of text difficulty and length.
  - User profile creation and login.
  - Storage and display of test history.
  - Leaderboard
  - Responsive design for various screen sizes.

- **2.3 Technologies Used**
  - Frontend: HTML, CSS, JavaScript.
  - Backend: PHP.
  - Database: MySQL

- **2.4 Timeline**
  - Phase 1 (1 week): Project Planning and System Design.
  - Phase 2 (2 weeks): Backend Development and Database Integration.
  - Phase 3 (3 week): Testing and Debugging.
  - Phase 4 (4,5 week): Deployment and Documentation.

- **2.5 Resource Allocation**
  - Developers: Responsible for coding and implementation.
  - Testing Team: Responsible for quality assurance.
  - Project Manager: Responsible for timeline and task management.

## 3. System Design

- **3.1 Architecture**
  - For a simple test, a purely frontend application.
  - For a more complex application requiring user accounts and saved data, a client-server architecture.
  - Client-side (Frontend): Browser.
  - Server-side (Backend): Application server and database.

- **3.2 Database Design**
  - User Table: (userID, username, password, email).
  - Test Results Table: (resultID, userID, WPM, accuracy, timestamp, textSample).

- **3.3 User Interface (UI) Design**
  - **3.3.1 Wireframes**
    - Basic layout of the typing test interface.
    - Placement of text display, input area, and results.
    - Navigation elements for settings and history.
  - **3.3.2 Mockups**
    - Detailed visual representation of the UI.
    - Color schemes, fonts, and interactive elements.
    - Examples of different test scenarios and result displays.
  - **3.3.3 Style Guide**
    - Consistent design elements (colors, fonts, spacing).
    - Guidelines for UI components and interactions.
    - Ensuring a clean and professional look.

- **3.4 Functional Specifications**
  - Text generation: Randomly select text samples from a database or a predefined list.
  - Input handling: Capture user input in real-time.
  - WPM calculation: Count the number of correctly typed words per minute.
  - Accuracy calculation: Determine the percentage of correctly typed characters.
  - Timer: Track the elapsed time during the test.
  - Result display: Show WPM, accuracy, and errors after the test.
  - User accounts: Allow users to create accounts and save their test history.

# Implementation

**4.HTML Structure Implementation:-**

- 4.1 HTML Structure
- 4.2 CSS Styling
- 4.3 JavaScript Logic
- **4.4 PHP Backend: -**
  - 4.4.1 Server-Side Scripting
  - 4.4.2 Database Interaction
  - 4.4.3 User Authentication

5.Testing:

5.1 Unit Testing

5.2 Integration Testing

5.3 User Acceptance Testing (UAT)

5.4 Performance Testing

5.5 Cross-Browser Compatibility

6.Deployment:-

- 6.1 Server Configuration
- 6.2 Database Setup
- 6.3 Frontend Deployment
- 6.4 Backend Deployment
- 6.5 Domain Setup

7.Maintenance:-

- 7.1 Bug Fixing
- 7.2 Updates and Enhancements
- 7.3 Performance Monitoring
- 7.4 Security Audits
- 7.5 User Support

8.Future Enhancements

9.Conclusion

10.References:-

  - 10.1 External Libraries
  - 10.2 Online Resources

1. Introduction:

1.1 Project Overview

- The Typing Speed Test project is a web application designed to measure and improve
- typing speed and accuracy. It provides users with a platform to practice
- typing, assess their skills, and track their progress.
- This application is developed using a combination of HTML for structure, CSS for styling, JavaScript for interactivity, and optionally PHP for server-side functionality (such as storing user data or high scores).

1.2 Purpose and Goals

- The primary purposes of this project are:

  - To provide a user-friendly tool for individuals to test their typing speed and accuracy.
  - To offer a platform for practice and improvement of typing skills. To create a web application that is accessible, responsive, and easy to use. To implement a system for tracking user progress and performance.
  - To demonstrate full-stack web development capabilities (HTML, CSS, JavaScript, PHP).

**1.3 Target Audience**

  - Individuals who want to assess their current typing skills.
  - Users who want to improve their typing speed and accuracy for personal or professional reasons.
  - Students who need to enhance their typing abilities for academic work.
  - Job seekers who require proficient typing skills for certain positions.
  - Anyone interested in a simple and effective typing practice tool.

## 2. Project Planning

### 2.1 Scope

This project includes the development of a web-based typing speed test application. The core features include:

- Displaying a randomly generated text or a selection of predefined texts.
- Providing a text input area for users to type.
- Timing the duration of the test.
- Calculating typing speed in words per minute (WPM).
- Calculating typing accuracy.

The project scope may be expanded to include additional features in future iterations, such as:

- Multiple difficulty levels.
- Customizable test durations.
- Different text categories.
- User accounts with progress tracking.
- Leaderboards and social sharing.

### 2.2 Features

The key features of the typing speed test application are:

- **Typing Test:** Presents the user with a passage of text to type.
- **Timer:** Measures the time taken by the user to complete the test.
- **Input Area:** Provides a space for the user to type the displayed text.
- **Speed Calculation:** Calculates the user's typing speed in words per minute (WPM). WPM is generally calculated as: (Number of Typed Characters / 5) / Time in Minutes.
- **Accuracy Calculation:** Calculates the accuracy of the user's typing by comparing the typed text with the original text. Accuracy is generally calculated as: (Number of Correct Characters / Total Number of Typed Characters) * 100.
- **Results Display:** Shows the user their typing speed and accuracy after the test.
- **Text Selection:** Offers a variety of pre-loaded texts, or the option for random text generation.
- **Responsive Design:** Adapts the layout to different screen sizes and devices.
- **User Accounts:** Allows users to create accounts, save their progress, and track their improvement over time.
- **Data Storage:** Stores user data, including test results and progress, in a database.
- **Leaderboards:** Displays top scores and rankings.

### 2.3 Technologies Used

The following technologies are used in this project:

- **Frontend:**
  - HTML: For structuring the web page.
  - CSS: For styling the web page and creating a responsive layout.
  - JavaScript: For implementing the interactive functionality of the typing test, including the timer, speed calculation, and results display.

- **Backend:**
  - PHP: For server-side scripting, handling user data, and database interaction.
  - MySQL: For database management.

- **Tools:**
  - Text editor (e.g., VS Code, Sublime Text, Atom).
  - Browser developer tools.
  - Database management tool (e.g., phpMyAdmin).

### 2.4 Timeline

The estimated timeline for completing the project is as follows:

- **Phase 1: Planning and Design (1 week)**
    - Requirements gathering and analysis.
    - Scope definition.

**Phase 2: Development (2-3 weeks)**

- Frontend development (HTML, CSS, JavaScript).
- Backend development (PHP, MySQL).
- Integration of frontend and backend.

**Phase 3: Testing (1 week)**

- Unit testing.
- Integration testing.
- User acceptance testing (UAT).
- Performance testing.

**Phase 4: Deployment (1 week)**

- Server setup and configuration.
- Database deployment.
- Frontend deployment.
- Final testing and deployment.

**Phase 5: Documentation (1 week)**

- Writing of all documentation.

**Total Estimated Time: 5-7 weeks**

The timeline is subject to change based on the complexity of the features and any unforeseen issues that may arise.

### 2.5 Resource Allocation

The resources required for this project include:

- **Development Team:**
  - 1-2 Frontend Developers.
  - 1 Backend Developer.
- **Software:**
  - Operating System (Windows)
  - Text editor.
  - Web browser.
  - Database management system.
- **Hardware:**
  - Computers for development.
- **Time:**
  - Dedicated time for each phase of the project.

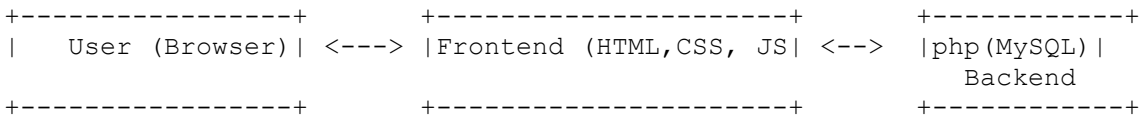
## 3. System Design

### 3.1 Architecture

The architecture of the typing speed test application can be described as follows:

- **Frontend (Client-Side):**
  - The user interface is built using HTML, CSS, and JavaScript.
  - JavaScript handles user interactions, updates the UI dynamically, and calculates typing speed and accuracy.
  - The frontend communicates with the backend using HTTP requests.
- **Backend (Server-Side):**
  - The backend is built using PHP.
  - PHP handles server-side logic, such as processing user data, interacting with the database, and managing user authentication.
  - The backend uses a database (e.g., MySQL) to store user data, test results, and other information.
- **Database:**
  - The database stores information such as:
    - User accounts (if implemented).
    - Test results (speed, accuracy, timestamp).
    - Predefined texts for the test.

**Architecture Diagram:**



### 3.2 Database Design

**Tables:**

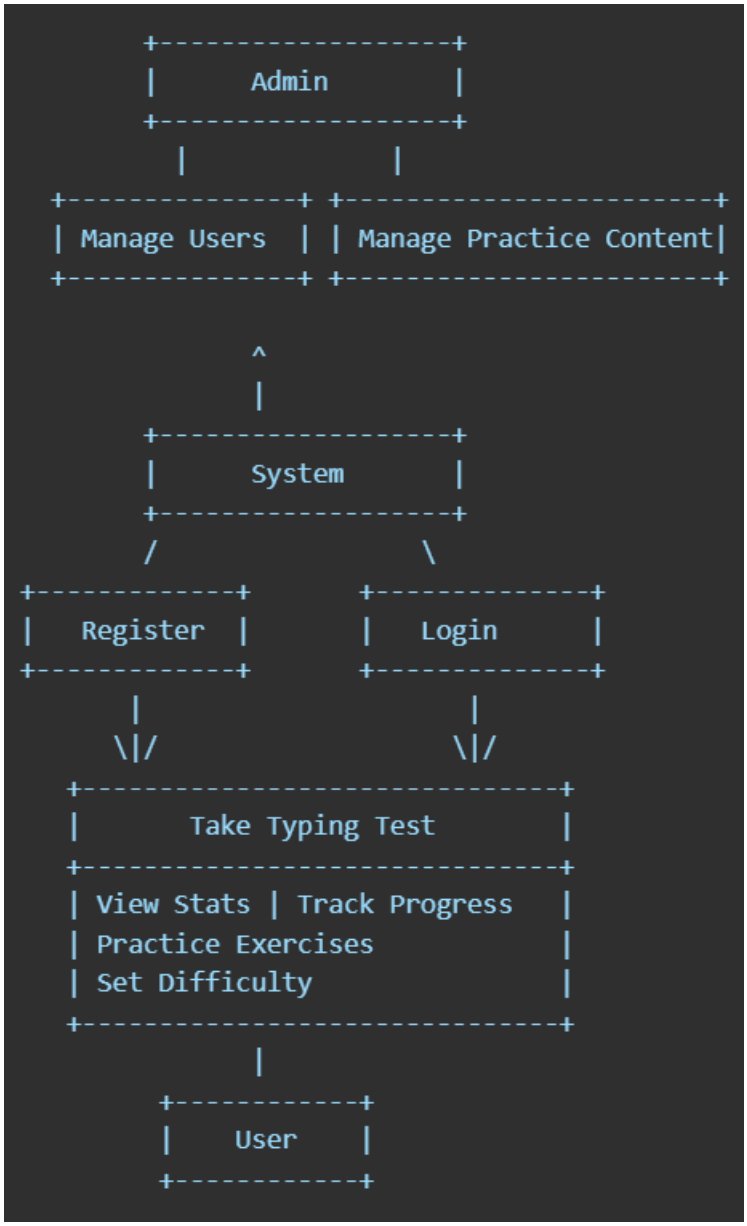
- **users:**
  - user_id (INT, PRIMARY KEY, AUTO_INCREMENT)
  - username (VARCHAR, UNIQUE, NOT NULL)
  - password (VARCHAR, NOT NULL)
  - email (VARCHAR, UNIQUE, NOT NULL)
  - created_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)

- **tests:**
  - test_id (INT, PRIMARY KEY, AUTO_INCREMENT)
  - user_id (INT, FOREIGN KEY referencing users.user_id)
  - text_id (INT, FOREIGN KEY referencing texts.text_id)
  - speed (INT, NOT NULL)
  - accuracy (DECIMAL, NOT NULL)
  - timestamp (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- **texts:**
  - text_id (INT, PRIMARY KEY, AUTO_INCREMENT)
  - content (TEXT, NOT NULL)
  - difficulty (VARCHAR)

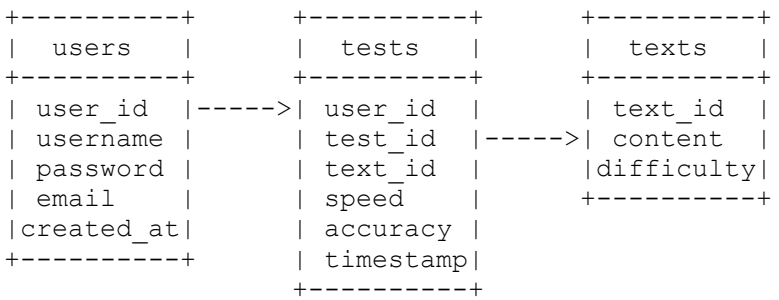
**Relationships:**

- A user can take multiple tests (one-to-many relationship between users and tests).
- A test is associated with one user (one-to-one relationship between tests and users).
- A test uses one text passage (one-to-one relationship between tests and texts).
- A text passage can be used in multiple tests (one-to-many relationship between texts and tests).

## Use Case Diagram:-



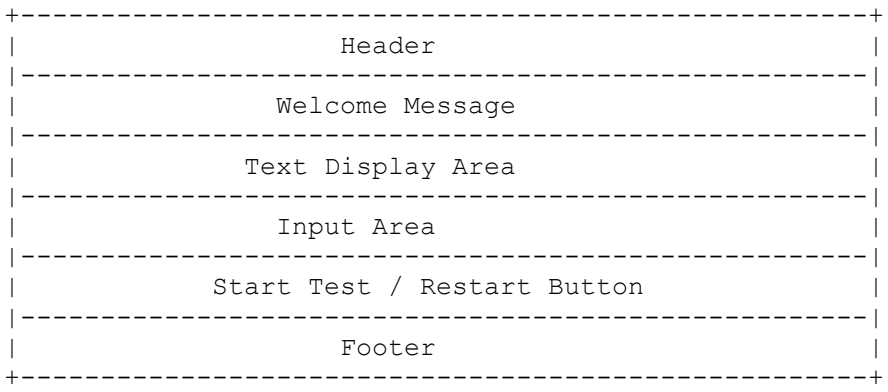

## class Diagram:



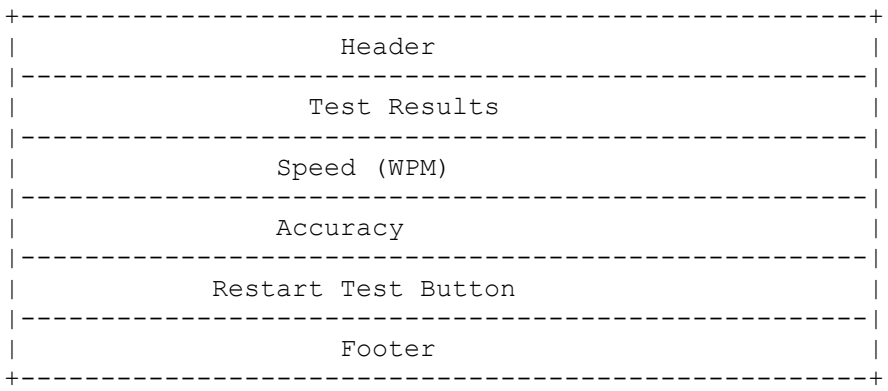
### 3.3.1 Wireframes

Wireframes are basic visual representations of the user interface, showing the layout of the elements on the page.

**Home Page Wireframe:**




**Results Page Wireframe:**



### 3.3.2 Mockups

Mockups are more detailed visual representations of the UI, including colors, fonts, and specific design elements. (Detailed mockups would be included here, showing the visual design of the application. Since I can't create images, I'll describe the design.)

**Design Description:**

- **Overall Style:** Clean, modern, and user-friendly. Use of a light color palette with contrasting text for readability.
- **Header:** A simple header with the application title (e.g., "Typing Speed Test") and possibly a navigation menu (if user accounts are implemented).
- **Text Display Area:** A prominent area where the text to be typed is displayed. Use of a clear, readable font (e.g., Arial, Times New Roman, or a modern sans-serif font). Sufficient padding and contrast to make the text easy to read.
- **Input Area:** A text input field where the user types. Styled to be visually distinct from the text display area.
- **Start/Restart Button:** A large, clearly labeled button to start or restart the test. Use of color and hover effects to indicate interactivity.
- **Results Display:** A section that displays the user's typing speed (WPM) and accuracy after the test is completed. Use of bold text and clear labels to highlight the results.
- **Footer:** A simple footer with copyright information or links.

### 3.3.3 Style Guide

A style guide defines the visual elements of the application, ensuring consistency across all pages.

**Colors:**

- Primary color: #3498db (Blue)
- Secondary color: #e74c3c (Red)
- Background color: #ecf0f1 (white and Dark)
- **Text color:** #2c3e50 (black)
- **Accent color:** #f1c40f (Blue)

**Fonts:**

- **Primary font:** Arial, sans-serif
- **Secondary font:** Times New Roman, serif

**Typography:**

- **Heading 1 (H1):** 24px, bold
- **Heading 2 (H2):** 20px, bold
- **Paragraph:** 16px, normal

**Buttons:**

- **Padding:** 10px 20px
- **Font size:** 16px
- **Background color:** #3498db
- **Border radius:** 5px
- **Hover effect:** Darker shade of blue

**Input Fields:**

- **Padding:** 10px
- **Font size:** 16px
- **Border:** 1px solid #bdc3c7
- **Border radius:** 5px

**Layout:**

- **Maximum width:** 1200px
- **Margin:** Auto (for centering)
- **Padding:** 20px

### 3.4 Functional Specifications

The functional specifications outline how the application should work.

- **Typing Test:**
  - The application should display a randomly selected text passage in the text display area.
  - The user should be able to type the text in the input area.
  - The application should start a timer when the user begins typing.
  - The application should record the time taken by the user to complete the test.
- **Timer:**
  - The timer should start when the user begins typing in the input area.
  - The timer should display the elapsed time or the remaining time.
  - The timer should stop when the user completes typing the text.
- **Input Area:**
  - The input area should be a text input field where the user can type.
  - The input area should be clear when the test starts or restarts.
  - The input area should allow the user to type the text.

- **Speed Calculation:**
  - The application should calculate the user's typing speed in words per minute (WPM).
  - WPM should be calculated by dividing the number of typed words by the time taken in minutes. A "word" is typically considered to be 5 characters.

## Typing Speed Test Application Overview

- **Accuracy Calculation:**
  - The application should calculate the accuracy of the user's typing.
  - Accuracy should be calculated by comparing the user's input with the original text and determining the percentage of correctly typed characters.
- **Results Display:**
  - After the user completes the test, the application should display the following:
    - Typing speed (WPM).
    - Typing accuracy (percentage).
    - The time taken to complete the test.
  - The results should be displayed in a clear and easy-to-read format.
- **Text Selection:**
  - The application should provide a selection of pre-loaded text passages for the user to type.
  - The application should provide an option to generate random text.
- **Responsive Design:**
  - The application should be responsive and adapt to different screen sizes, including desktops, tablets, and smartphones.
- **User Accounts:**
  - Users should be able to create accounts and log in.
  - User accounts should store user information, such as username, email, and password.
  - Users should be able to view their profile and test history.
- **Data Storage:**
  - The application should store user data, including test results, in a database.
  - Data should be stored securely and efficiently.
- **Leaderboards:**
  - The application should display a leaderboard showing the top scores.
  - Users should be able to see their ranking on the leaderboard.

## Implementation

### *HTML Structure*

The HTML structure of the typing speed test application consists of the following elements:

```
<!DOCTYPE html>
<html lang="en" data-theme="dark">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Typing Speed Test</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css">
</head>
<body>
    <div class="theme-toggle">
        <button id="theme-toggle-btn">
            <i class="fas fa-sun"></i>
        </button>
    </div>

    <div class="container home-container">
        <header>
            <h1>Typing Speed Test</h1>
            <p class="tagline">Improve your typing speed and accuracy</p>
        </header>

        <div class="features">
            <div class="feature-card">
                <i class="fas fa-tachometer-alt"></i>
                <h3>Speed Test</h3>
                <p>Test your typing speed with different difficulty
levels</p>
            </div>
            <div class="feature-card">
                <i class="fas fa-chart-line"></i>
                <h3>Track Progress</h3>
                <p>Monitor your improvement over time</p>
            </div>
            <div class="feature-card">
                <i class="fas fa-trophy"></i>
                <h3>Leaderboard</h3>
```

```
                <p>Compete with other typists</p>
            </div>
        </div>

        <?php if (isset($_SESSION['user_id'])): ?>
            <div class="difficulty-selector">
                <h2>Choose Your Challenge</h2>
                <div class="difficulty-cards">
                    <a href="typing_test.php?difficulty=easy"
class="difficulty-card">
                        <i class="fas fa-star"></i>
                        <h3>Easy</h3>
                        <p>Short, simple sentences</p>
                        <span class="difficulty-level">Beginner
Friendly</span>
                    </a>
                    <a href="typing_test.php?difficulty=medium"
class="difficulty-card">
                        <i class="fas fa-star"></i>
                        <i class="fas fa-star"></i>
                        <h3>Medium</h3>
                        <p>Longer sentences with common words</p>
                        <span class="difficulty-level">Intermediate</span>
                    </a>
                    <a href="typing_test.php?difficulty=hard"
class="difficulty-card">
                        <i class="fas fa-star"></i>
                        <i class="fas fa-star"></i>
                        <i class="fas fa-star"></i>
                        <h3>Hard</h3>
                        <p>Complex paragraphs with technical content</p>
                        <span class="difficulty-level">Expert Level</span>
                    </a>
                </div>
            </div>
        <?php else: ?>
            <div class="auth-buttons">
                <a href="login.php" class="btn btn-primary">Login</a>
                <a href="register.php" class="btn btn-
secondary">Register</a>
            </div>
        <?php endif; ?>
    </div>

    <script>
```

```
        // Theme toggle functionality
        const themeToggleBtn = document.getElementById('theme-toggle-
btn');
        const html = document.documentElement;

        // Check for saved theme preference
        const savedTheme = localStorage.getItem('theme') || 'dark';
        html.setAttribute('data-theme', savedTheme);
        updateThemeIcon(savedTheme);

        themeToggleBtn.addEventListener('click', () => {
            const currentTheme = html.getAttribute('data-theme');
            const newTheme = currentTheme === 'light' ? 'dark' : 'light';

            html.setAttribute('data-theme', newTheme);
            localStorage.setItem('theme', newTheme);
            updateThemeIcon(newTheme);
        });

        function updateThemeIcon(theme) {
            const icon = themeToggleBtn.querySelector('i');
            icon.className = theme === 'light' ? 'fas fa-moon' : 'fas fa-
sun';
        }
    </script>
</body>
</html>
```

Explanation of HTML Elements

- **`<!DOCTYPE html>`**: Declares the document type as HTML5.
- **`<html>`**: The root element of the HTML page.
- **`<head>`**: Contains meta-information about the HTML document, such as character set, viewport settings, title, and links to CSS files. It also includes the `<script src="script.js"></script>` to load the JavaScript.
- **`<body>`**: Contains the visible content of the HTML page.
- **`<header>`**: Contains the header section of the page, including the title of the application.
- **`<main>`**: Contains the main content of the page, including the text display area, input area, timer, results area, and start/restart button.
- **`<div id="text-display-area">`**: Contains the text that the user needs to type.
- **`<div id="input-area">`**: Contains the `<textarea>` element where the user types.
- **`<div id="timer-area">`**: Displays the timer.
- **`<div id="results-area">`**: Contains the results of the typing test (speed and accuracy). Initially hidden using `style="display: none;"`.
- **`<button id="start-button">`** / **`<button id="restart-button">`**: Buttons to start or restart the test.

### Footer and JavaScript Link

- **`<footer>`**: Contains the footer section of the page, including copyright information.
- **`<script src="script.js"></script>`**: Links the JavaScript file to the HTML. Best practice is to include this before the closing `</body>` tag for performance reasons.

## 4.2 CSS Styling

The CSS is used to style the HTML elements and create a visually appealing and responsive user interface.

```
/* Reset default styles */
:root {
    --primary-color: #2563eb;
    --secondary-color: #1e40af;
    --background-color: #ffffff;
    --text-color: #000000;
    --text-secondary: #4b5563;
    --border-color: #e5e7eb;
    --shadow-color: rgba(0, 0, 0, 0.1);
    --success-color: #059669;
    --error-color: #dc2626;
    --card-bg: #f8fafc;
    --input-bg: #ffffff;
}

[data-theme="dark"] {
    --primary-color: #3b82f6;
    --secondary-color: #1e40af;
    --background-color: #111827;
    --text-color: #f3f4f6;
    --text-secondary: #9ca3af;
    --border-color: #374151;
    --shadow-color: rgba(0, 0, 0, 0.3);
    --success-color: #10b981;
```

```
    --error-color: #ef4444;
    --card-bg: #1f2937;
    --input-bg: #374151;
}

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: var(--background-color);
    color: var(--text-color);
    line-height: 1.6;
    transition: background-color 0.3s, color 0.3s;
}

.theme-toggle {
    position: fixed;
    top: 20px;
    right: 20px;
    cursor: pointer;
    z-index: 1000;
    font-size: 1.5rem;
    color: var(--text-color);
}

header {
    background-color: var(--secondary-color);
    padding: 1rem 2rem;
    box-shadow: 0 2px 5px var(--shadow-color);
}

nav {
    display: flex;
    justify-content: space-between;
    align-items: center;
    max-width: 1200px;
    margin: 0 auto;
}

.logo {
    font-size: 1.5rem;
```

```
    font-weight: bold;
    color: var(--primary-color);
}

.nav-links a {
    color: var(--text-color);
    text-decoration: none;
    margin-left: 2rem;
    transition: color 0.3s;
    font-weight: 500;
}

.nav-links a:hover {
    color: var(--primary-color);
}

main {
    max-width: 1200px;
    margin: 2rem auto;
    padding: 0 1rem;
}

section {
    display: none;
}

section.active {
    display: block;
}

/* Welcome Section */
#welcome-section {
    text-align: center;
    padding: 4rem 0;
}

#welcome-section h1 {
    font-size: 2.5rem;
    margin-bottom: 1rem;
    color: var(--primary-color);
}

.start-practice-btn {
    background-color: var(--primary-color);
    color: white;
```

```
    border: none;
    padding: 1rem 2rem;
    font-size: 1.2rem;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
    margin-top: 2rem;
}

.start-practice-btn:hover {
    background-color: #357abd;
}

/* Practice Section */
.practice-container {
    background-color: var(--background-color);
    padding: 2rem;
    border-radius: 10px;
    box-shadow: 0 0 20px var(--shadow-color);
}

.stats-bar {
    display: flex;
    justify-content: space-around;
    margin-bottom: 2rem;
    padding: 1rem;
    background-color: var(--secondary-color);
    border-radius: 5px;
    color: white;
}

.stat {
    text-align: center;
}

.text-display {
    background-color: var(--background-color);
    padding: 2rem;
    border: 2px solid var(--border-color);
    border-radius: 5px;
    margin-bottom: 1rem;
    min-height: 150px;
    font-size: 1.2rem;
    line-height: 1.8;
}
```

- **`*`**: Resets default margins and padding for all elements.
- **`body`**: Sets the font, background color, and text color for the entire page. Uses flexbox to center content vertically and horizontally.
- **`header`**: Styles the header section, centering the title.
- **`main`**: Sets the maximum width of the main content area and centers it.
- **`#text-display-area`**: Styles the area where the test text is displayed. Key features include background color, border, padding, text alignment, and `white-space: pre-wrap;` to preserve line breaks.
- **`#input-area`**: Styles the input area, centering the textarea.
- **`#text-input`**: Styles the textarea where the user types. Sets the width, padding, font size, border, and disables resizing.
- **`#timer-area`**: Styles the timer display.
- **`#results-area`**: Styles the results area, including background, border, padding, and text alignment. It's initially hidden using `display: none;` in the HTML.
- **`button`**: Styles the buttons (start and restart). Includes hover effects.
- **`footer`**: Styles the footer.
- **`@media (max-width: 600px)`**: A media query that applies specific styles for screens smaller than 600px, making the layout responsive for mobile devices. Font sizes and padding are adjusted for smaller screens.

### 4.3 JavaScript Logic

The JavaScript code implements the interactive functionality of the typing speed test.

```
// Typing Practice Configuration
const config = {
    timeLimit: 60, // seconds
    easyWords: ['the', 'be', 'to', 'of', 'and', 'a', 'in', 'that', 'have',
'I', 'it', 'for', 'not', 'on', 'with', 'he', 'as', 'you', 'do', 'at'],
    mediumWords: ['computer', 'programming', 'development', 'software',
'database', 'network', 'security', 'application', 'interface', 'system'],
    hardWords: ['algorithm', 'encryption', 'authentication',
'optimization', 'architecture', 'implementation', 'deployment',
'maintenance', 'integration', 'configuration']
};

const API_URL = 'http://localhost/typing-platform/api'; // Update this
with your actual API URL

// DOM Elements
const textDisplay = document.getElementById('text-display');
const inputField = document.getElementById('input-field');
const startBtn = document.getElementById('start-btn');
const resetBtn = document.getElementById('reset-btn');
const difficultySelect = document.getElementById('difficulty-select');
const wpmDisplay = document.getElementById('wpm');
const accuracyDisplay = document.getElementById('accuracy');
const timerDisplay = document.getElementById('timer');

// State Variables
let currentText = '';
let timeLeft = config.timeLimit;
let timer = null;
let isTyping = false;

// Text Generation
function generateText(difficulty) {
    const wordList = config[`${difficulty}Words`];
    const words = [];
    const wordCount = 20;

    for (let i = 0; i < wordCount; i++) {
```

```
        const randomIndex = Math.floor(Math.random() * wordList.length);
        words.push(wordList[randomIndex]);
    }

    return words.join(' ');
}

// Timer Function
function updateTimer() {
    timeLeft--;
    timerDisplay.textContent = `${timeLeft}s`;

    if (timeLeft === 0) {
        endTyping();
    }
}

// Start Typing
function startTyping() {
    if (isTyping) return;

    const difficulty = difficultySelect.value;
    currentText = generateText(difficulty);
    textDisplay.textContent = currentText;
    inputField.value = '';
    inputField.disabled = false;
    inputField.focus();
    timeLeft = config.timeLimit;
    isTyping = true;

    timer = setInterval(updateTimer, 1000);
    startBtn.textContent = 'Restart';
}

// Reset Typing
function resetTyping() {
    clearInterval(timer);
    isTyping = false;
    inputField.value = '';
    inputField.disabled = true;
    textDisplay.textContent = '';
    timeLeft = config.timeLimit;
    timerDisplay.textContent = `${timeLeft}s`;
    wpmDisplay.textContent = '0';
    accuracyDisplay.textContent = '100%';
```

```
    startBtn.textContent = 'Start';
}

// Calculate WPM
function calculateWPM() {
    const wordsTyped = inputField.value.trim().split(/\s+/).length;
    const minutes = (config.timeLimit - timeLeft) / 60;
    return Math.round(wordsTyped / minutes);
}

// Calculate Accuracy
function calculateAccuracy() {
    const correctChars = [...inputField.value].filter((char, index) =>
char === currentText[index]).length;
    const totalChars = inputField.value.length;
    return Math.round((correctChars / totalChars) * 100) || 100;
}

// Save Results to Backend
async function saveResults(wpm, accuracy, difficulty, wordsTyped) {
    try {
        const response = await fetch(`${API_URL}/save_result.php`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${localStorage.getItem('token')}`
            },
            body: JSON.stringify({
                wpm,
                accuracy,
                difficulty,
                words_typed: wordsTyped
            })
        });

        if (!response.ok) {
            throw new Error('Failed to save results');
        }

        const data = await response.json();
        console.log('Results saved:', data);
    } catch (error) {
        console.error('Failed to save results:', error);
    }
}
```

- **Variables**: Declare variables to store references to HTML elements, timer-related data, test status, and results.
- **`generateText()`**: Selects a random text passage from an array and displays it in the `#test-text` element.
- **`startTimer()`**: Initializes the timer, sets the initial time, and starts an interval that updates the timer display every second. It also sets the `startTime` variable.
- **`calculateSpeed()`**: Calculates the typing speed in words per minute (WPM). It divides the number of correct characters by 5 (characters per word) and then divides by the elapsed time in minutes.
- **`calculateAccuracy()`**: Calculates the typing accuracy as a percentage of correctly typed characters to total typed characters.
- **`endTest()`**: Stops the timer, calculates the speed and accuracy, displays the results, and disables the input area.
- **`startTest()`**: Generates new text, clears the input area, hides the results, starts the timer, and enables the input area.
- **`textInputElement.addEventListener('input', ...)`**: An event listener that is triggered whenever the user types in the input area. It compares the typed text with the expected text, updates the `correctChars` and `totalChars` counts, and optionally applies styling for correct and incorrect characters.
- **`startButton.addEventListener('click', startTest)`**: Starts the test when the user clicks the "Start Test" button.
- **`restartButton.addEventListener('click', startTest)`**: Restarts the test when the user clicks the "Restart Test" button.
- **Initial setup**: The code calls `generateText()` when the page loads to display the initial text. It also disables the text input until the user clicks the start button.

## 4.4 PHP Backend (if applicable)

If you want to store user data (e.g., test results, user accounts), you'll need a backend using PHP and a database (e.g., MySQL). This section provides a basic outline.

### *4.4.1 Server-Side Scripting*

PHP scripts will handle requests from the frontend, process data, and interact with the database. For example:

- **`save_result.php`**: Saves the user's test results to the database.
- **`get_history.php`**: Retrieves the user's test history from the database.
- **`register.php`**: Registers a new user account.
- **`login.php`**: Authenticates a user and starts a session.
- **`logout.php`**: Logs a user out and destroys the session.

### *4.4.2 Database Interaction*

PHP scripts will use the `mysqli` or `PDO` extension to connect to the MySQL database and perform queries. For example:-

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "typing_speed_db";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

### *4.4.3 User Authentication*

For user accounts, you'll need to implement user authentication using sessions and password hashing.

- **Registration**: When a user registers, their password should be hashed using a function like `password_hash()` before being stored in the database.
- **Login**: When a user logs in, the entered password should be hashed and compared to the hashed password stored in the database using `password_verify()`. If the passwords match, a session should be started.
- **Session Management**: Sessions should be used to store user data (e.g., user ID) and maintain the user's logged-in state.
- **Logout**: The session should be destroyed when the user logs out.

## 5. Testing

Testing is crucial to ensure the application functions correctly and meets the requirements.

### *5.1 Unit Testing*

Unit testing involves testing individual components or functions of the application in isolation. For example:

- Testing the `calculateSpeed()` function with different inputs to ensure it returns the correct speed.
- Testing the `calculateAccuracy()` function with different inputs to ensure it returns the correct accuracy.
- Testing the `generateText()` function to ensure it returns a valid text string.

### *5.2 Integration Testing*

Integration testing involves testing how different components of the application work together. For example:

- Testing the interaction between the input area and the timer to ensure the timer starts correctly when the user begins typing.
- Testing the interaction between the input area and the results display to ensure the results are displayed correctly after the test is completed.
- Testing the frontend and backend integration to ensure data is correctly sent and received.

### *5.3 User Acceptance Testing (UAT)*

User acceptance testing involves testing the application with real users to ensure it meets their needs and expectations. This may involve:

- Having users perform typical tasks, such as taking a typing test and viewing their results.
- Gathering feedback from users on the usability and design of the application.
- Identifying any issues or bugs that users encounter.

### *5.4 Performance Testing*

Performance testing involves evaluating the application's speed, responsiveness, and stability under different conditions. For example:

- Testing the application with a large number of concurrent users.
- Measuring the time it takes for the application to load.
- Identifying any performance bottlenecks.

### *5.5 Cross-Browser Compatibility*

Cross-browser compatibility testing ensures that the application works correctly in different web browsers (e.g., Chrome, Firefox, Safari, Edge). This involves:

- Testing the application in different browsers to ensure the layout and functionality are consistent.
- Identifying and fixing any browser-specific issues.

## 6. Deployment

Deployment is the process of making the application available to users.

### *6.1 Server Configuration*

If you are using a backend (PHP), you will need to configure a web server (e.g., Apache, Nginx) to host the application. This involves:

- Installing and configuring the web server software.
- Setting up virtual hosts for the application.
- Configuring server settings, such as file permissions and security.

### *6.2 Database Setup*

If you are using a database (MySQL), you will need to set up the database and import the database schema. This involves:

- Installing and configuring the database server.
- Creating the database.
- Creating the tables (e.g., using SQL scripts).
- Setting up database users and permissions.

### *6.3 Frontend Deployment*

The frontend (HTML, CSS, JavaScript) can be deployed by:

- Uploading the files to a web server.
- Using a content delivery network (CDN) to host the static assets.

### *6.4 Backend Deployment*

The backend (PHP) can be deployed by:

- Uploading the PHP files to the web server.
- Ensuring the PHP interpreter is correctly configured.

Configuring the application to connect to the database.

### 6.5 Domain Setup

To make the application accessible through a domain name (e.g., www.example.com), you will need to:

- Register a domain name.
- Configure DNS records to point the domain name to the server's IP address.

7. Maintenance

- Maintenance is the ongoing process of keeping the application running smoothly and addressing any issues that may arise.

7.1 Bug Fixing

- Bug fixing involves identifying and resolving any errors or defects in the application. This may involve:

- Reproducing the bug.
- Identifying the cause of the bug.
- Developing and testing a fix.
- Deploying the fix to the production environment.

7.2 Updates and Enhancements :

- Updates and enhancements involve adding new features, improving existing features, and keeping the application up-to-date with the latest technologies. This may involve:

- Gathering user feedback and feature requests.
- Planning and designing new features.
- Developing and testing the new features.
- Deploying the updates to the production environment.

7.3 Performance Monitoring

- Performance monitoring involves tracking the application's performance over time to identify any potential issues. This may involve:

- Monitoring server load, response times, and error rates.
- Using tools to analyze application performance.
- Optimizing the application to improve performance.

7.4 Security Audits

- Security audits involve assessing the application's security to identify any vulnerabilities. This may involve:

- Performing regular security scans.
- Reviewing the code for potential security flaws.
- Implementing security best practices.

### 7.5 User Support

- User support involves providing assistance to users who encounter problems or have questions about the application. This may involve:

- Providing documentation and FAQs.
- Responding to user inquiries via email or other channels.
- Troubleshooting user issues.

## 8. Future Enhancements :

- Future enhancements for the typing speed test application could include:

- User accounts with progress tracking: Allow users to track their typing speed and accuracy over time.
- Multiple difficulty levels: Offer different text passages with varying levels of difficulty.
- Different text categories: Provide texts from different categories, such as literature, technical, or news.
- Customizable test durations: Allow users to choose the duration of the typing test.
- Leaderboards and social sharing: Implement leaderboards to show top scores and allow users to share their results on social media.
- Advanced statistics: Provide more detailed statistics, such as words per minute for different sections of the text, and error rates for specific keys.
- Typing games: Add game elements to make practicing typing more engaging.
- Support for multiple languages: Allow users to test their typing speed in different languages.
- Personalized training plans: Offer customized training plans based on the user's performance.

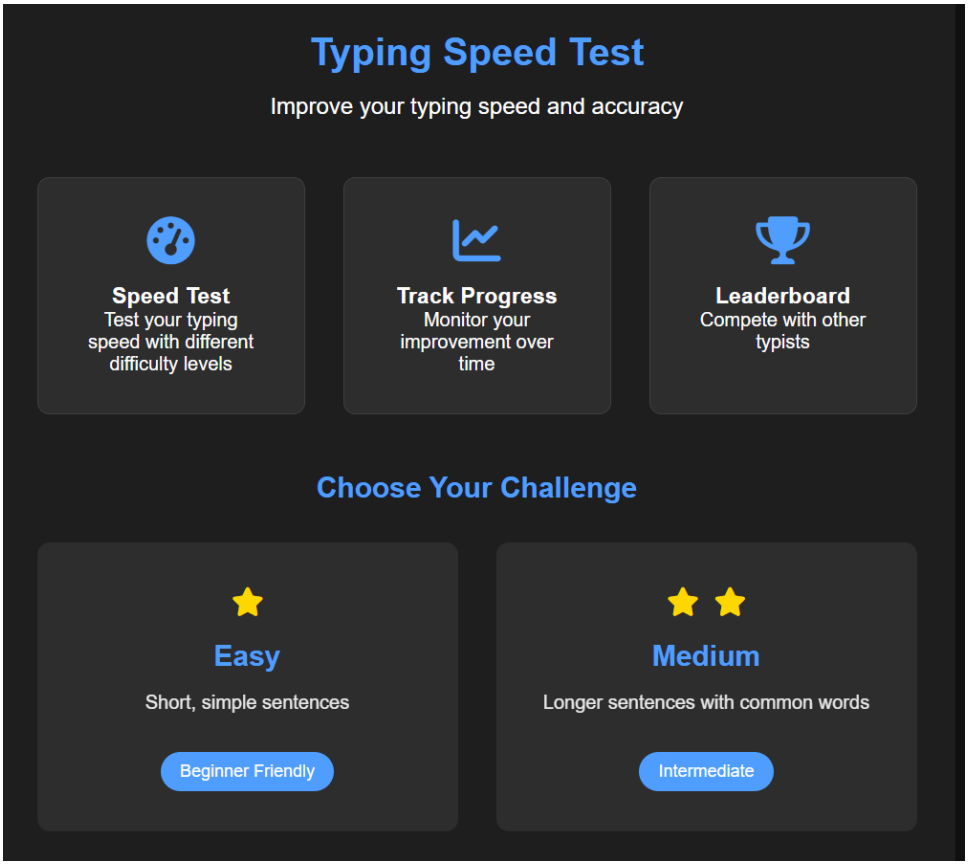
9. Conclusion:

The Typing Speed Test project provides a valuable tool for individuals to assess and improve their typing skills. The application is designed to be user-friendly, responsive, and accurate. With the optional backend implementation, it can also provide features for user accounts, data storage, and progress tracking. The project demonstrates a solid understanding of web development technologies and best practices. Future enhancements can further expand the functionality and user engagement of the application.

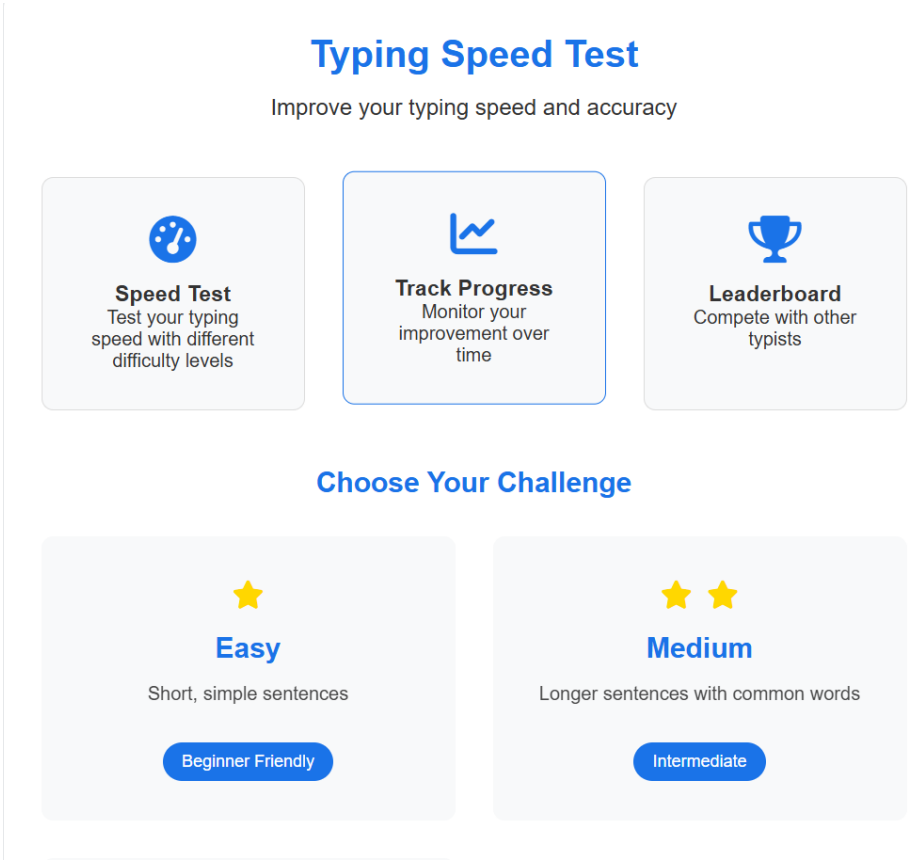
10. References :

10.1 External Libraries I will put the project pictures now:-

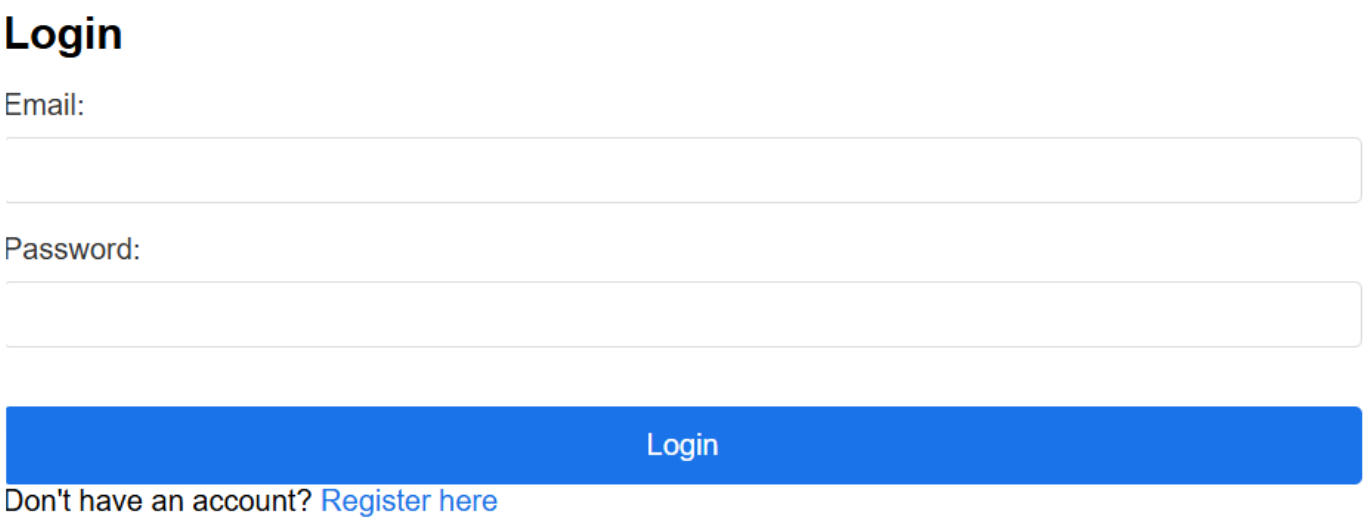
## Dark page:-



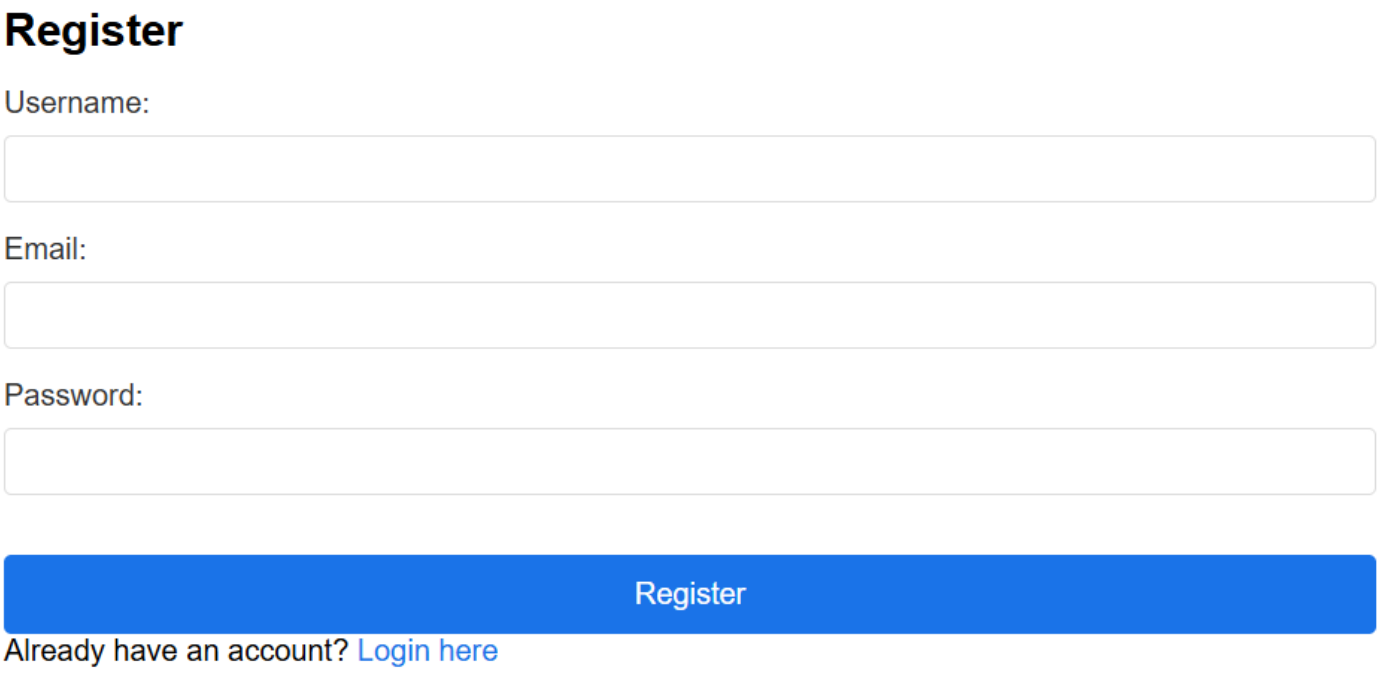
# The bright page :-



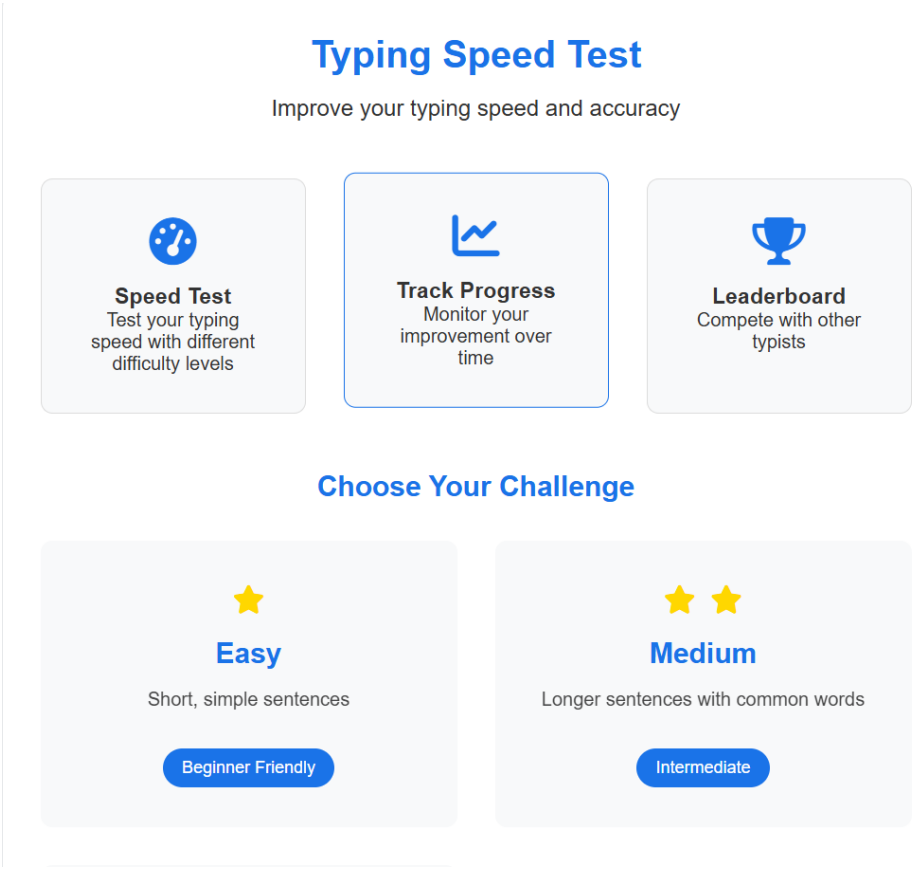
**login page:-**


Login
Email:
Password:
Login
Don't have an account? Register here

**Register Page:-**



# The Home:-


Typing Speed Test
Improve your typing speed and accuracy
Speed Test
Test your typing speed with different difficulty levels
Track Progress
Monitor your improvement over time
Leaderboard
Compete with other typists
Choose Your Challenge
Easy
Short, simple sentences
Beginner Friendly
Medium
Longer sentences with common words
Intermediate

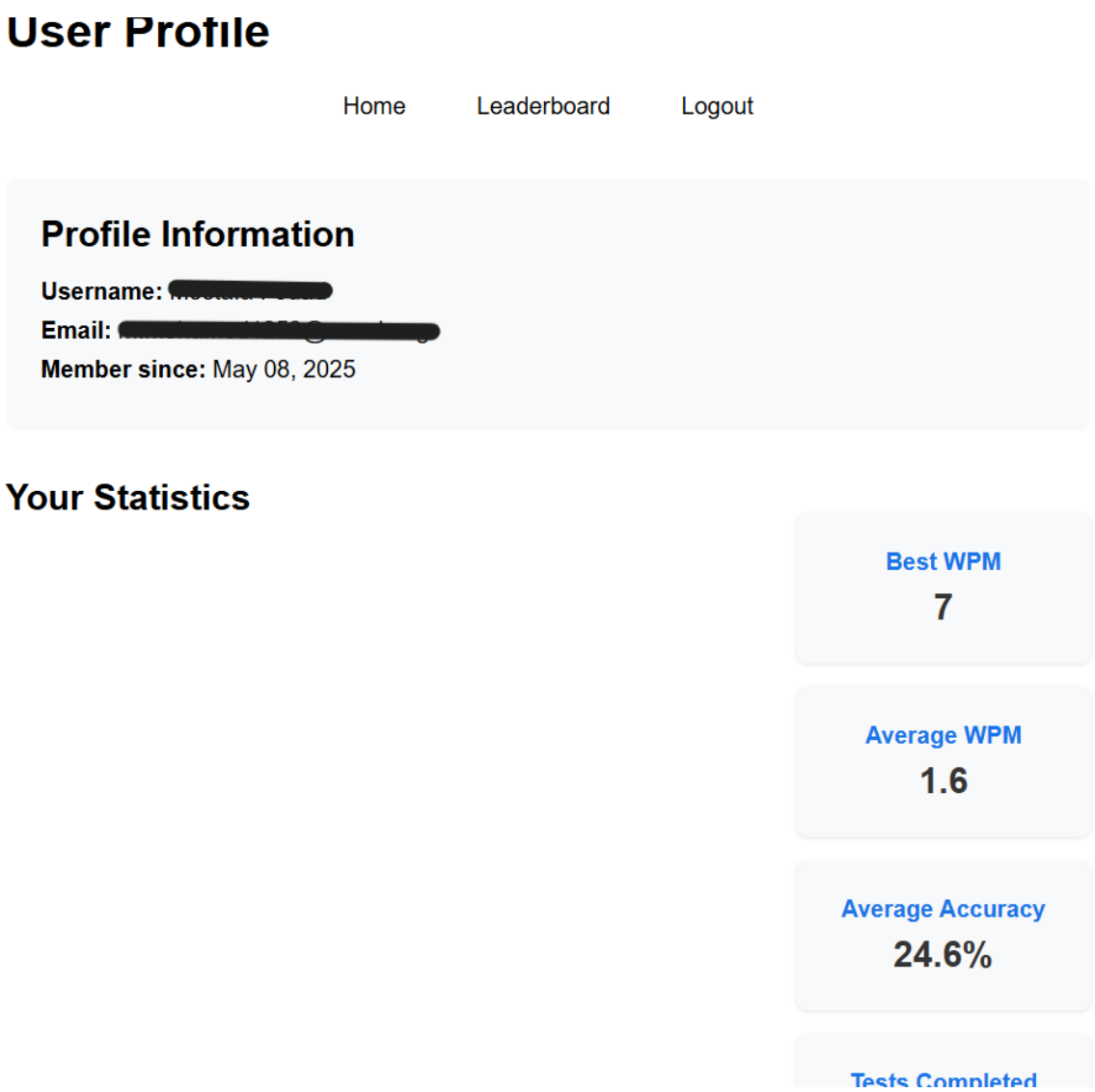
## The Leaderboard :-

# Typing Speed Leaderboard

Home Profile Logout

| Rank | Username | WPM | Accuracy | Date |
|---|---|---|---|---|
| 1 | [redacted] | 7 | 83% | May 08, 2025 |
| 2 | [redacted] | 3 | 100% | May 08, 2025 |
| 3 | [redacted] | 2 | 6% | May 08, 2025 |
| 4 | [redacted] | 1 | 7% | May 08, 2025 |
| 5 | [redacted] | 1 | 5% | May 08, 2025 |
| 6 | [redacted] | 1 | 2% | May 08, 2025 |
| 7 | [redacted] | 0 | 0% | May 08, 2025 |
| 8 | [redacted] | 0 | 0% | May 08, 2025 |
| 9 | [redacted] | 0 | 0% | May 08, 2025 |

**The profile:-**




10.2 Online Resources :-

- MDN Web Docs: https://developer.mozilla.org/en-US/1-
- W3Schools: https://www.w3schools.com/-
- (List any other relevant online resources)3-