

Objective:

The primary objective of this project, *Typing Master Pro*, is to design and develop an interactive typing game that helps users enhance their typing speed, accuracy, and overall efficiency. In the modern digital world, fast and accurate typing is an essential skill for students, professionals, and anyone who uses computers regularly. This project aims to create a platform where users can not only practice but also track their performance over time, making the learning process engaging and motivating.

Through the use of structured HTML, stylish CSS, and functional JavaScript, this project delivers a seamless and responsive user experience. The interface is kept clean and minimalistic to ensure users can focus entirely on their typing without distractions. Features like dynamic word changes, real-time speed and accuracy calculation, and interactive elements such as the "Start Game" button, contribute to keeping the user engaged throughout their practice sessions.

Moreover, the project seeks to demonstrate good coding practices such as separation of concerns, clear and maintainable code, and proper commenting for better understanding. With responsive design techniques, it ensures compatibility across various devices, making it accessible for users on desktops, tablets, and smartphones alike.

An optional yet valuable objective of this project is to integrate advanced enhancements like animations, loading spinners, and performance graphs (if needed). These elements not only make the application visually appealing but also provide a smooth and professional feel to the overall experience.

Ultimately, the goal is to create an educational tool that is easy to use, quick to load, and effective in helping users improve their typing skills, making practice both productive and enjoyable. Whether used for personal development or educational purposes, this typing game serves as a practical and rewarding application of web development concepts.

Introduction

In the modern digital era, typing has become an essential skill for everyone, whether they are students, professionals, or casual computer users. Fast and accurate typing saves time and boosts productivity in tasks such as writing emails, coding, creating documents, or chatting online. To help people improve their typing skills in an engaging and interactive way, we have developed a web-based application called *Typing Master Pro*. This project is designed to combine learning with fun by providing users with an exciting platform to practice and enhance their typing speed and accuracy.

The main objective of *Typing Master Pro* is to create an engaging and responsive typing game that challenges users to type displayed words as quickly and accurately as possible. At the end of the game, users receive instant feedback in the form of statistics, including words per minute (WPM) and accuracy percentage. These metrics help users track their performance and encourage continuous improvement. The project also serves as a great demonstration of web development skills by making effective use of HTML, CSS, and JavaScript to build a fully functional and user-friendly application.

We chose this project because typing games offer a practical and enjoyable way to practice essential skills. At the same time, it allows us to explore and implement important concepts of front-end development such as event handling, DOM manipulation, game logic, animations, and responsive design. This project not only strengthens our technical abilities but also enhances our understanding of how to create interactive web applications that provide real value to users.

In this project, HTML5 is used to build the structure of the web page, CSS3 adds attractive styling and layout, and JavaScript powers the game logic and user interaction. The design is kept simple and clean to ensure a smooth user experience. Features like responsive design ensure that the game works well on different devices, while animations and transitions make the game visually appealing.

Overall, *Typing Master Pro* is not just a game, but a learning tool that helps users practice typing skills effectively. Through this project, we have gained hands-on experience in web development, improved our problem-solving abilities, and learned the importance of writing clean and maintainable code. This project also opens up opportunities for future improvements such as adding new difficulty levels, a leaderboard, or multiplayer functionality. It has been a valuable learning experience and an enjoyable way to apply our technical knowledge in a practical setting.

HTML(Structure):

In the development of the Typing Master project, HTML (HyperText Markup Language) plays a fundamental role as it is responsible for creating the basic structure and layout of the web pages. HTML provides the framework upon which CSS and JavaScript add style and interactivity.

Let's break it down in detail:

1. Page Layout and Structure

The overall structure of the page is created using HTML.

We first create the skeleton of the web page, which defines:

- Where elements like headings, paragraphs, and buttons will appear.
- Logical flow of the content from top to bottom.
- Proper use of containers to group related elements.

For example:

```
html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Typing Master </title>
</head>
<body>
  <div class="container">
    <h1>Welcome to Typing Master Pro</h1>
    <p>Test and improve your typing speed!</p>
    <button id="startButton">Start Game</button>
  </div>
</body>
</html>
```

- Here, <div> is used as a container to group the heading, paragraph, and button.
- The <h1> tag is used for the main heading.

- The `<p>` tag is used for descriptive text.
- The `<button>` provides an action point for the user.

2. Using Semantic Tags

Semantic HTML tags give meaning to the web page rather than just presentation. This helps both browsers and developers understand the role of each part of the web page.

In this project:

- `<h1>` represents the main title or heading of the page.
- `<p>` is used to display supporting text or description.
- `<button>` is an interactive element that allows the user to start the game.
- `<div>` acts as a generic container to hold all these elements together.

Benefits of using semantic tags:

- Better accessibility
- SEO (Search Engine Optimization) friendly
- Easier for screen readers to interpret
- Cleaner and more organized code

Example:

```
html
<div class="container">
  <h1>Typing Master </h1>
  <p>Sharpen your typing skills with this fun game!</p>
  <button id="startButton">Start Game</button>
</div>
```

3. Linking CSS and JavaScript Files

To make the page look attractive and interactive, we link external CSS and JavaScript files.

This follows the principle of Separation of Concerns, where:

- HTML handles structure,
- CSS handles presentation,
- JavaScript handles behavior.

How to link CSS:

```
html
<head>
```

```
<link rel="stylesheet" href="style.css">
</head>
```

How to link JavaScript:

```
html
<body>
  <script src="script.js"></script>
</body>
```

By doing this, we keep our code clean and maintainable. Any changes to the design or functionality can be done directly in their respective files without modifying the HTML structure.

4. Button to Start the Game

The most important interactive element on the page is the Start Game button.

This button is created using the `<button>` tag and is given a unique id so it can be accessed and controlled via JavaScript.

```
html
<button id="startButton">Start Game</button>
```

- The `id="startButton"` allows JavaScript to recognize this specific button and add functionality.
- When the user clicks the button, JavaScript captures the event and redirects them to the game page using:

```
javascript
document.getElementById('startButton').addEventListener('click', function() {
  window.location.href = 'main.html';
});
```

This makes the page interactive and starts the typing game seamlessly.

CSS (Design & Styling):

CSS (Cascading Style Sheets) is responsible for the look and feel of our Typing Master Pro project. While HTML gives the structure, CSS transforms that structure into a visually appealing and user-friendly interface.

Let's explore the important aspects of CSS used in this project:

1. Styling Layout and Typography

CSS allows us to control:

- Font style and size
- Colors

- Spacing (margin, padding)
- Backgrounds
- Layout positioning

For example, in our project, we use CSS to:

- Set the font to a clean and readable style like Arial or sans-serif.
- Add color to headings and buttons.
- Adjust margins and padding for a better visual balance.

Sample CSS:

```
css
body {
  font-family: 'Arial', sans-serif;
  color: #fff;
  background-color: #1e1e2f;
  margin: 0;
  padding: 0;
}

h1 {
  font-size: 2.5rem;
  margin-bottom: 10px;
}

p {
  font-size: 1.2rem;
  margin-bottom: 20px;
}
```

Typography is important because it improves readability and user experience. Proper font size and spacing make the game welcoming and easy to use.

2. Flexbox for Centering Elements

Flexbox is one of the most powerful layout models in CSS.

It helps align elements perfectly in the center of the screen, both vertically and horizontally.

In this project, we use Flexbox to center the content inside the container:

css

```
.container {  
  display: flex;  
  justify-content: center; /* Horizontal centering */  
  align-items: center; /* Vertical centering */  
  height: 100vh; /* Full height of viewport */  
  flex-direction: column; /* Stack elements vertically */  
}
```

Advantages of Flexbox:

- Responsive layouts become easier.
- Minimal code for powerful alignment.
- Adapts well to different screen sizes.

By using Flexbox, we ensure that whether the user opens the game on a laptop, tablet, or mobile, the layout stays centered and clean.

3. Gradients for Attractive Background

A plain background can look boring, so we use CSS gradients to create more visual interest.

In our project, we apply a linear gradient as the background:

css

```
body {  
  background: linear-gradient(135deg, #667eea, #764ba2);  
}
```

Explanation:

- 135deg sets the direction of the gradient.
- #667eea and #764ba2 are color codes that blend smoothly.

Benefits:

- Makes the game look modern and professional.
- Gradients are lightweight and do not require image files.
- Enhances visual engagement without slowing down the page.

4. Responsive Design

Today, users open websites on different devices.

Responsive design ensures that the website looks good on:

- Mobile phones
- Tablets
- Desktops

We use media queries in CSS to change styles based on screen size:

```
css
@media (max-width: 600px) {
  h1 {
    font-size: 1.8rem;
  }
  p {
    font-size: 1rem;
  }
  button {
    padding: 10px 20px;
  }
}
```

With this:

- Text and buttons adjust their size.
- Layout adapts to smaller screens.
- The user experience remains smooth across all devices.

Responsive design is very important for accessibility and user satisfaction.

5. Hover Effects and Transitions for Button Interactivity

To make our buttons interactive and lively, we add hover effects and smooth transitions.

Example:

```
css
button {
  background-color: #4caf50;
  color: white;
  padding: 15px 30px;
  border: none;
  border-radius: 5px;
  font-size: 1rem;
```

```
cursor: pointer;

transition: background-color 0.3s ease, transform 0.2s ease;
}

button:hover {
  background-color: #45a049;
  transform: scale(1.05);
}
```

Explanation:

- transition makes changes (color, transform) happen smoothly.
- On hover, the button changes color and slightly enlarges.
- This provides visual feedback to the user, making the interface feel responsive and alive.

Benefits:

- Better user experience.
- Helps guide the user's attention.
- Makes the game feel interactive and modern.

JavaScript (Functionality):

JavaScript is the brain of our Typing Master Pro game.

While HTML provides structure and CSS gives style, JavaScript brings functionality and interactivity to life!

It controls how the game behaves, how the user interacts, and how the game responds to user actions.

Let's explore the key JavaScript concepts used in this project:

1. DOM Manipulation

DOM (Document Object Model) is a programming interface that allows JavaScript to access and change the content, structure, and style of web pages.

When we write HTML, it is converted into the DOM by the browser.

Using JavaScript, we can:

- Change the text on the page
- Update styles dynamically
- Add or remove elements

Example in our project:

```
javascript
```



```
const wordDisplay = document.getElementById('wordDisplay');
wordDisplay.textContent = randomWord;
```

Explanation:

- `document.getElementById()` selects an element by its ID.
- `.textContent` changes the text inside that element.

Other useful methods:

- `document.querySelector()`
- `document.createElement()`
- `element.style.property = value`

Benefits:

- Makes the page dynamic.
- Updates content without reloading the page.
- Provides live feedback to users (like score, time left, etc.).

2. Event Handling (`addEventListener`)

In our typing game, we need to respond to user actions — such as typing, clicking the start button, or submitting words.

JavaScript uses event listeners to detect these actions.

Example:

```
javascript
startButton.addEventListener('click', startGame);
```

Explanation:

- `addEventListener()` attaches an event (like 'click') to an element.
- When the button is clicked, the `startGame` function runs.

More examples:

```
javascript
document.addEventListener('keydown', handleKeyPress);
```

Common events:

- `click` — when user clicks
- `keydown` — when a key is pressed
- `input` — when user types in a field
- `submit` — when a form is submitted

Benefits:

- Makes the page interactive.
- Reacts instantly to user input.
- Provides a better user experience.

3. Functions and Methods

Functions are blocks of code designed to perform a particular task.

In our game, functions help to:

- Start the game
- Generate random words
- Update the timer
- Track the score

Example:

```
javascript
function startGame() {
    score = 0;
    time = 60;
    userInput.value = "";
    generateWord();
    updateTimer();
}
```

Explanation:

- Functions group code logically.
- They can be reused multiple times.
- Improve readability and maintenance.

Methods are functions associated with objects.

Example:

```
javascript
console.log('Game started!');
wordList.push('newword');
```

Benefits:

- Makes the code modular.
- Helps in debugging and testing.
- Improves code organization.

4. Page Redirection (`window.location.href`)

In our project, after the game ends, we may want to:

- Redirect the user to another page
- Show results on a new screen
- Navigate to an index page

For this, we use:

```
javascript
window.location.href = 'results.html';
```

Explanation:

- `window.location.href` sets the URL of the page.
- It navigates to the specified address.

Example use in the project:

```
javascript
if (time <= 0) {
  clearInterval(timer);
  window.location.href = 'result.html';
}
```

Benefits:

- Smooth navigation between pages.
- Improves user flow.
- Helps organize the project into multiple pages.

Other useful methods:

- `window.open()`
- `window.history.back()`

5. `DOMContentLoaded` Event to Initialize Functions

Sometimes, our JavaScript runs before the HTML is fully loaded, causing errors.

To avoid this, we use:

```
javascript
document.addEventListener('DOMContentLoaded', function() {
  initializeGame();
});
```

Explanation:

- DOMContentLoaded fires when the HTML is completely loaded and parsed.
- Ensures our functions run only after the page is ready.

In our project:

- We can safely set up event listeners.
- Initialize variables and UI elements.
- Avoid errors from missing elements.

Benefits:

- Safe and reliable initialization.
- Prevents runtime errors.
- Improves loading performance.

Project Structuring:

When we build any web-based project, it's very important to keep the project well-organized and clean. Project structuring helps in easy understanding, future modifications, and teamwork.

Our Typing Master Pro project follows these principles for better readability and maintenance.

Let's understand how:

1. Separation of Concerns (HTML, CSS, JS in Different Files)

Definition:

Separation of concerns means keeping different technologies in separate files based on their responsibility:

- HTML — Structure of the web page
- CSS — Styling and design
- JavaScript (JS) — Functionality and logic

Why do we do this?

- Keeps the code organized.
- Easier to read and understand.
- Allows multiple people to work on the project (one person can work on CSS, another on JS).

In our project:

- index.html → Structure of the game (layout, buttons, headings)
- style.css → Colors, fonts, layout styling, and animations
- script.js → Game logic, timers, scoring, and interactivity

Example:

index.html

html

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

```
<script src="script.js"></script>
```

By linking CSS and JS like this, the files stay separate and organized.

Benefits:

- Cleaner project structure.
- Easy debugging and troubleshooting.
- Scalability for future updates.

2. Clean, Maintainable Code

Definition:

Clean code is easy to read, understand, and maintain.

Maintainable code allows future developers (or yourself!) to easily modify or upgrade the project.

How we maintain clean code:

- Consistent formatting (indentation, spacing)
- Logical grouping of functions and styles
- Avoiding unnecessary code duplication
- Using modern coding practices

Example:

Clean Function Example:

javascript

```
function generateRandomWord() {  
    const randomIndex = Math.floor(Math.random() * words.length);  
    return words[randomIndex];  
}
```

Avoiding messy code like this:

javascript

```
function rand() { return words[Math.floor(Math.random() * words.length)]; }
```

Benefits:

- Easy to update in the future.
- Understandable for others reading your code.
- Professionalism in coding habits.

3. Commenting and Proper Naming Conventions

Definition:

Adding comments and using meaningful names in code makes it easier to understand the purpose of each part of the code.

Why is this important?

- Helps others (or yourself) understand what each part does.
- Makes it easier to debug and update the code later.
- Increases readability and documentation.

Commenting Example:

```
javascript
// Start the game and initialize variables
function startGame() {
    score = 0;
    time = 60;
    userInput.value = "";
    generateWord();
    updateTimer();
}
```

Naming Conventions Example:

Good:

```
javascript
const userInputField = document.getElementById('userInput');
const startGameButton = document.getElementById('startButton');
```

Bad:

```
javascript
const x = document.getElementById('userInput');
const y = document.getElementById('startButton');
```

Tips for good comments and names:

- Use descriptive variable and function names.
- Write comments to explain complex logic.
- Keep comments updated as code changes.

Benefits:

- Easier collaboration with teammates.
- Faster problem-solving and debugging.
- Professional-quality project documentation.

User Experience (UX):

User Experience (UX) refers to the overall experience of a user when interacting with our application. A good UX design makes the user feel comfortable, engaged, and satisfied while using the application.

In our Typing Master Pro, we have focused on creating a simple, fast, and user-friendly experience. Let's understand the key points:

1. Simple and Clean Interface

Definition:

A simple and clean interface means:

- No clutter.
- Easy-to-read fonts.
- Proper use of space.
- Clear instructions.

Why is this important?

- Helps users focus on the game.
- Avoids confusion.
- Makes the application look professional.

In Our Project:

- We have a centered layout to keep attention on the game.
- Used large headings and readable fonts for clarity.
- Displayed the timer, score, and words in visible positions.

Example:

```
html
<h1>Typing Master Pro</h1>
<p>Type the words as fast as you can!</p>
<div id="wordDisplay">randomWord</div>
```

Benefits:

- Better focus for the user.
- Cleaner look and feel.

- Reduced distractions.

2. Call to Action (Start Game Button)

Definition:

A Call to Action (CTA) is an element (usually a button) that tells users what to do next.

Importance of CTA:

- Directs user flow.
- Makes the application interactive.
- Encourages immediate engagement.

In Our Project:

- We added a clear, visible "Start Game" button.
- Positioned centrally for easy access.
- Styled it with hover effects for better interactivity.

Example:

```
html
<button id="startButton">Start Game</button>
```

CSS Styling:

```
css
#startButton {
  background-color: #4CAF50;
  color: white;
  padding: 15px 30px;
  font-size: 18px;
  border-radius: 10px;
  cursor: pointer;
}
#startButton:hover {
  background-color: #45a049;
}
```

Benefits:

- Clear user direction.
- Encourages users to start playing quickly.
- Improves user engagement.

3. Fast Loading and Responsive Design

Definition:

- Fast Loading: Ensures the page loads quickly, providing a smooth experience.
- Responsive Design: Adjusts the layout to work well on all screen sizes (mobile, tablet, desktop).

Importance:

- Prevents user frustration.
- Works well across devices.
- Improves accessibility.

In Our Project:

- Used minimalistic code for faster loading.
- Used flexbox to center elements for responsive layout.
- Used media queries (if needed) to adjust the layout on different screen sizes.

Example:

Flexbox CSS:

```
css
body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}
```

Responsive Text:

```
css
h1 {
  font-size: 2.5em;
}
@media (max-width: 600px) {
  h1 {
    font-size: 1.8em;
  }
}
```

Benefits:

- Smooth and fast experience.
- Accessible on mobile devices.
- Reduces user frustration from slow loading.

Future Scope (Expanded):

The Typing Master Pro project holds great potential for future development and enhancements. Currently, it provides a solid foundation for improving typing skills, but many advanced features can be integrated to make the experience even more engaging, competitive, and educational.

In the future, the game can introduce features like multiplayer mode, where users can compete in real-time typing challenges with their friends or global users. Adding a user login system will allow players to save their progress, track improvement over time, and unlock achievements for motivation.

Additionally, AI-based performance analysis could be incorporated to give users personalized feedback on common typing mistakes and offer custom exercises to improve weak areas. A leaderboard system will also make the game more competitive by ranking users based on their typing speed and accuracy.

The user interface can be enhanced with more animations, visual effects, and even sound effects to provide instant feedback on correct and incorrect typing. A dark mode or customizable themes can improve accessibility and user comfort.

To attract a wider audience, the game could include multiple language support, helping users practice typing in different languages. For educational institutions, a teacher dashboard can be added to monitor student progress in classrooms.

Furthermore, mobile optimization or a dedicated mobile app version would allow users to practice anytime and anywhere. Finally, integrating API support for exporting results or sharing achievements on social media would make the game more connected and modern.

- Multiplayer mode for real-time competition.
- User login and profile system for progress tracking.
- AI-based personalized feedback and custom exercises.
- Global leaderboard for competitive engagement.
- Advanced animations and sound effects.
- Dark mode and customizable themes.
- Multi-language support for global users.
- Teacher dashboard for monitoring student progress.
- Mobile-friendly design or dedicated mobile app.
- Social media integration and result sharing.
- Cloud storage for saving user history.
- Offline mode for uninterrupted practice.

Limitations of the Project:

While the Typing Master Pro project successfully fulfills its primary goal of improving typing speed and accuracy, it still has certain limitations that can be addressed in future versions. Currently, the application operates as a basic, single-player typing game with limited customization and feedback options. It primarily focuses on speed measurement and lacks deeper analysis of typing errors, such as specific key weaknesses or error patterns.

Moreover, the game is dependent on an internet browser environment and is not yet optimized for all screen sizes or mobile devices. This limits its accessibility for users who prefer practicing on smartphones or tablets. The absence of user authentication also means that progress tracking is not personalized, and users cannot save or resume their practice sessions.

The design and user interface, while clean and functional, can be further enhanced to provide a more engaging and interactive user experience. Currently, features like detailed performance history, real-time competition, and advanced animations are missing, which could otherwise make the game more dynamic and motivating.

Finally, the application does not yet offer multi-language support, which restricts its usability for non-English speakers who wish to practice typing in their native language. Integrating more diverse content and challenges would also help in catering to a wider audience.

- Limited to basic single-player mode.
- No advanced error analysis or personalized feedback.
- Lacks user login system and progress saving.
- Not fully optimized for mobile devices.
- No multi-language support.
- Basic design and limited animations.
- No real-time multiplayer or leaderboard.
- Limited customization options for users.
- Dependent on browser environment, no standalone app.
- Does not include voice assistance or accessibility features.
- No integration with external learning platforms.
- No offline mode for practice without internet.

Implementation:

The implementation of the **Typing Master Pro** project involves the integration of HTML, CSS, and JavaScript to create an interactive, user-friendly typing game. The development was done in a structured way, ensuring separation of concerns where each technology handled its dedicated role effectively.

.html

```
<!DOCTYPE html>

<html lang="en">

<head>
```

```
<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Typing Master Pro Extended</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet">

<style>

body {

  background: linear-gradient(to right, #6a11cb, #2575fc);

  font-family: 'Poppins', sans-serif;

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100vh;

  margin: 0;

  overflow: hidden;

}

.container {

  background: white;

  padding: 30px;

  border-radius: 20px;

  box-shadow: 0 15px 30px rgba(0,0,0,0.3);

  text-align: center;

  width: 95%;

  max-width: 900px;

  animation: fadeIn 1s ease-in-out;

}

@keyframes fadeIn {

  from { opacity: 0; transform: translateY(20px); }

  to { opacity: 1; transform: translateY(0); }

}

h1 {

  margin-bottom: 10px;

  color: #333;
```

```
font-weight: 600;
font-size: 2rem;
}
#word {
font-size: 24px;
margin: 20px 0;
color: #2575fc;
font-weight: bold;
min-height: 60px;
}
#word span.correct { color: green; }
#word span.incorrect { color: red; }
input {
padding: 15px;
width: 95%;
border: 2px solid #2575fc;
border-radius: 8px;
margin-bottom: 20px;
font-size: 16px;
}
.stats {
display: grid;
grid-template-columns: repeat(2, 1fr);
gap: 10px;
margin-bottom: 15px;
font-weight: 600;
font-size: 14px;
color: #333;
}
.options {
margin-bottom: 15px;
}
button {
```

```
padding: 12px 25px;
background: linear-gradient(to right, #6a11cb, #2575fc);
color: white;
border: none;
border-radius: 8px;
cursor: pointer;
font-weight: bold;
transition: transform 0.2s;
}
button:hover {
  transform: scale(1.05);
}
#progressBar {
  width: 100%;
  height: 10px;
  background: #ddd;
  border-radius: 5px;
  overflow: hidden;
  margin-bottom: 15px;
}
#progressBarFill {
  height: 100%;
  width: 100%;
  background: #6a11cb;
  transition: width 0.2s;
}
</style>
</head>
<body>
<div class="container">
  <h1>Typing Master 🚀 </h1>
  <div class="options">
    Mode:
```

```

<select id="mode">
  <option value="word">Word</option>
  <option value="line" selected>Line</option>
</select>

Difficulty:
<select id="difficulty">
  <option value="easy">Easy</option>
  <option value="medium" selected>Medium</option>
  <option value="hard">Hard</option>
</select>
</div>

<div id="word">Loading...</div>

<input type="text" id="textInput" placeholder="Start typing..." autofocus />

<div id="progressBar"><div id="progressBarFill"></div></div>

<div class="stats">
  <div>Time: <span id="time">30</span>s</div>
  <div>Score: <span id="score">0</span></div>
  <div>WPM: <span id="wpm">0</span></div>
  <div>Accuracy: <span id="accuracy">100</span>%</div>
</div>

<button onclick="startGame()">Restart Game</button>
</div>

<audio id="correctSound" src="https://assets.mixkit.co/sfx/preview/mixkit-positive-interface-beep-221.mp3"></audio>

```

```

<script>

const wordDisplay = document.getElementById('word');
const textInput = document.getElementById('textInput');
const scoreDisplay = document.getElementById('score');
const timeDisplay = document.getElementById('time');
const wpmDisplay = document.getElementById('wpm');
const accuracyDisplay = document.getElementById('accuracy');

```

```
const modeSelect = document.getElementById('mode');
const difficultySelect = document.getElementById('difficulty');
const progressBarFill = document.getElementById('progressBarFill');
const correctSound = document.getElementById('correctSound');

const words = ['function', 'variable', 'object', 'program', 'syntax', 'array', 'compile', 'debug',
'execute', 'interface', 'network', 'hardware', 'software', 'cloud', 'server', 'database', 'memory', 'storage',
'cache', 'binary'];

const lines = [
  'Consistent practice improves typing skills',
  'Cloud computing is the future of technology',
  'Debugging is twice as hard as writing code',
  'The database stores persistent data',
  'Learn algorithms to solve problems efficiently',
  'Network security is critical for data protection',
  'Software development requires patience and focus',
  'Interfaces define how components interact',
  'Always comment your code for better readability'
];

let currentText = "";
let score = 0;
let time = 30;
let timer;
let typedChars = 0;
let correctChars = 0;

function getRandomText() {
  const mode = modeSelect.value;
  const difficulty = difficultySelect.value;
  let list = mode === 'word' ? words : lines;
  if (difficulty === 'easy') list = list.slice(0, 5);
  else if (difficulty === 'medium') list = list.slice(0, 10);
```



```
    return list[Math.floor(Math.random() * list.length)];
}

function displayNewText() {
    currentText = getRandomText();
    wordDisplay.innerHTML = currentText.split("").map(char => `<span>${char}</span>`).join("");
    textInput.value = "";
}

function startGame() {
    clearInterval(timer);
    time = 30;
    score = 0;
    typedChars = 0;
    correctChars = 0;
    timeDisplay.textContent = time;
    scoreDisplay.textContent = score;
    wpmDisplay.textContent = 0;
    accuracyDisplay.textContent = 100;
    progressBarFill.style.width = '100%';
    displayNewText();
    textInput.focus();
    timer = setInterval(updateTime, 1000);
}

function updateTime() {
    time--;
    timeDisplay.textContent = time;
    progressBarFill.style.width = `${(time / 30) * 100}%`;
    if (time === 0) {
        clearInterval(timer);
        alert(`Time's up! Your score: ${score}`);
    }
}
```

```

}

textInput.addEventListener('input', () => {
  const input = textInput.value;
  const spans = wordDisplay.querySelectorAll('span');
  typedChars++;

  input.split("").forEach((char, idx) => {
    if (char === currentText[idx]) {
      spans[idx].classList.add('correct');
      spans[idx].classList.remove('incorrect');
    } else {
      spans[idx]?.classList.add('incorrect');
      spans[idx]?.classList.remove('correct');
    }
  });

  correctChars = input.split("").filter((char, idx) => char === currentText[idx]).length;

  if (input === currentText) {
    score++;
    scoreDisplay.textContent = score;
    correctSound.play();
    displayNewText();
  }

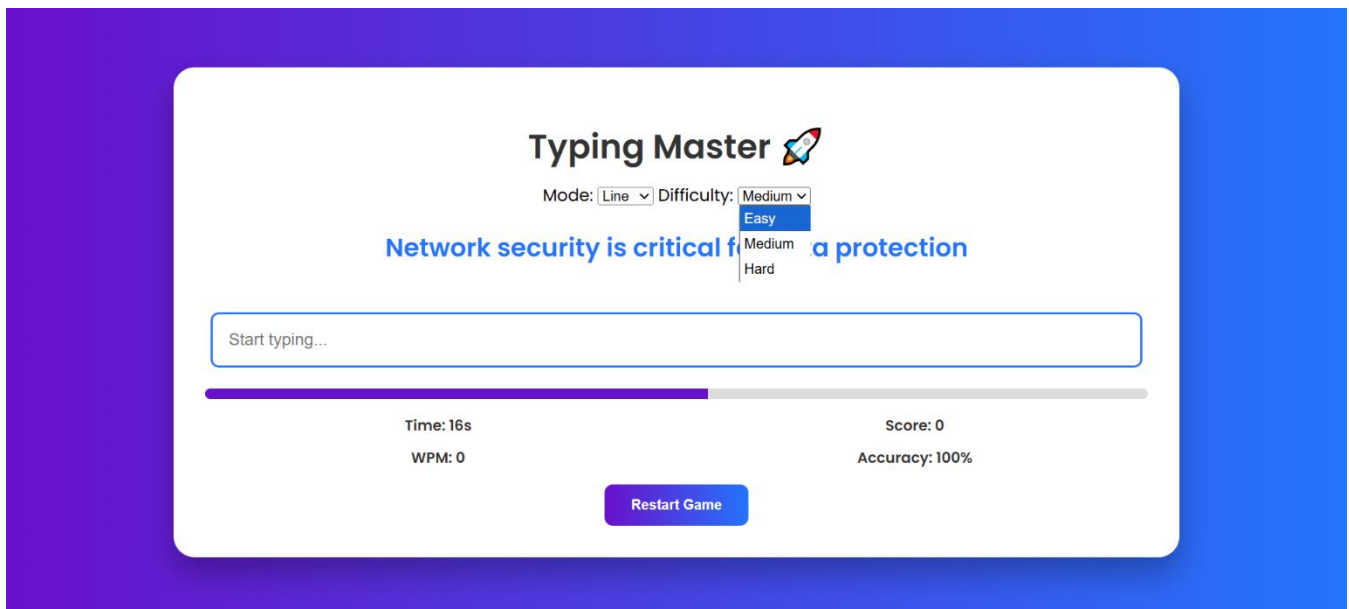
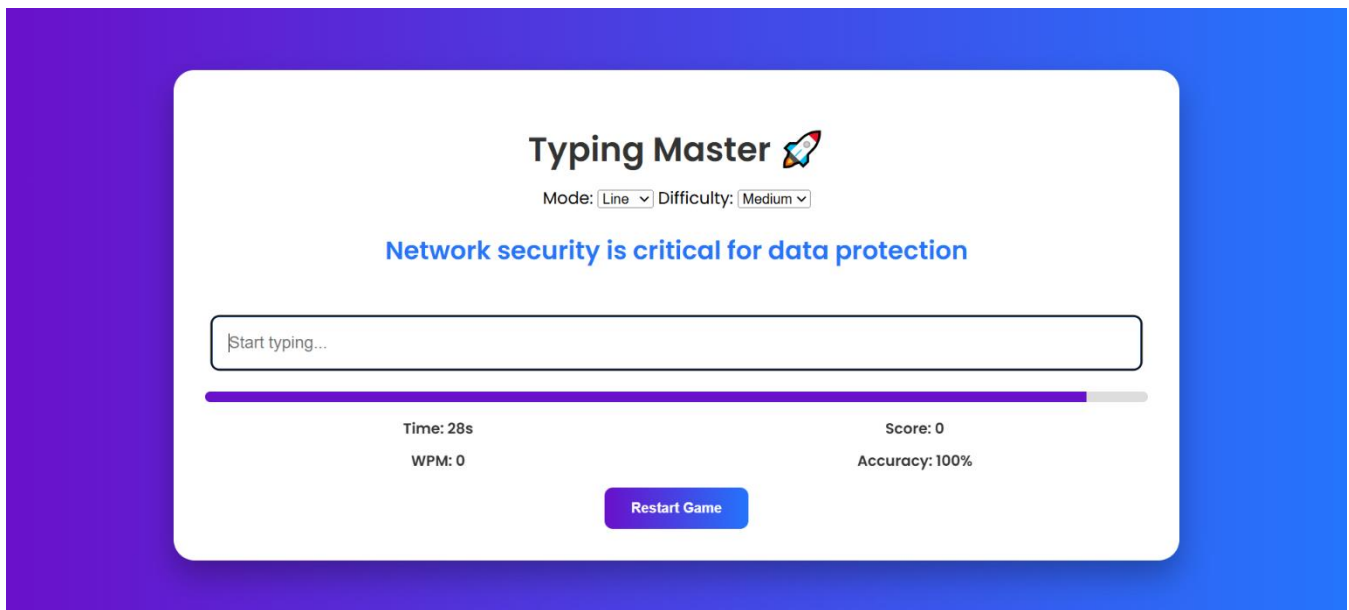
  const accuracy = typedChars > 0 ? Math.round((correctChars / typedChars) * 100) : 100;
  const wpm = Math.round((score * (60 / (30 - time || 1))) * (modeSelect.value === 'line' ? 4 : 1));
  accuracyDisplay.textContent = accuracy;
  wpmDisplay.textContent = wpm;
});


startGame();

```

```
</script>
</body>
</html>
```

Output:



Typing Master 

Mode: Line Difficulty: Medium

Easy

Medium

Hard

Network security is critical for a protection

Start typing...


Time: 18s

WPM: 0

Score: 0

Accuracy: 100%

Restart Game

Typing Master 

Mode: Line Difficulty: Medium

Easy

Medium

Hard

Network security is critical for a protection

Start typing...


Time: 16s

WPM: 0

Score: 0

Accuracy: 100%

Restart Game

Typing Master 

Mode: Line Difficulty: Medium

Easy

Medium

Hard

Network security is critical for a protection

Start typing...


Time: 14s

WPM: 0

Score: 0

Accuracy: 100%

Restart Game

Typing Master 

Mode: Line Difficulty: Medium

Easy

Medium

Hard

Network security is critical for a protection

Start typing...


Time: 10s

WPM: 0

Score: 0

Accuracy: 100%

Restart Game

Typing Master 

Mode: Line Difficulty: Medium

Learn algorithms to solve problems efficiently

Start typing...


Time: 29s

WPM: 0

Score: 0

Accuracy: 100%

Restart Game

Typing Master 

Mode: Line Difficulty: Medium

Learn algorithms to solve problems efficiently

learn algorithms to solve problems efficiently

Time: 5s


WPM: 0

Score: 0

Accuracy: 85%

Restart Game


Type here to search



Air quality forecast

21:24

06-04-2025

Typing Master 

Mode: Word Difficulty: Medium

object

Start typing...


Time: 23s

WPM: 20

Score: 2

Accuracy: 60%

Restart Game

Typing Master 

Mode: Word Difficulty: Medium

array

Start typing...


Time: 19s

WPM: 18

Score: 3

Accuracy: 29%

Restart Game

Typing Master 

Mode: Word Difficulty: Medium

execute

Start typing...

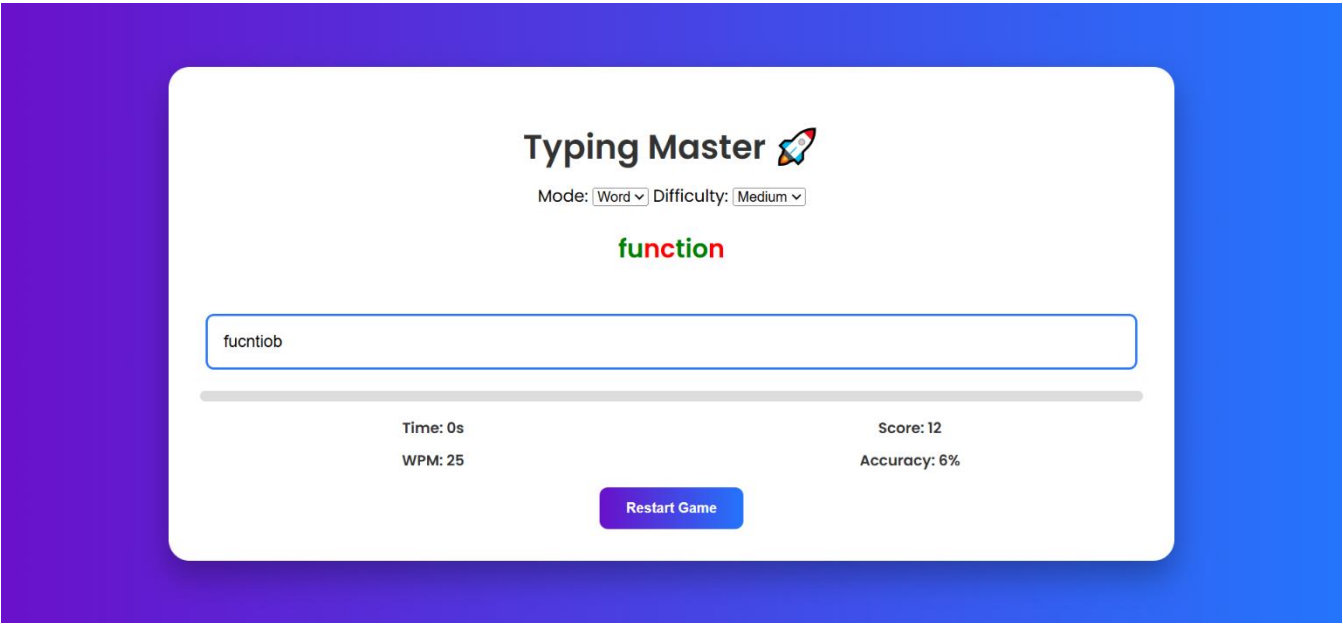
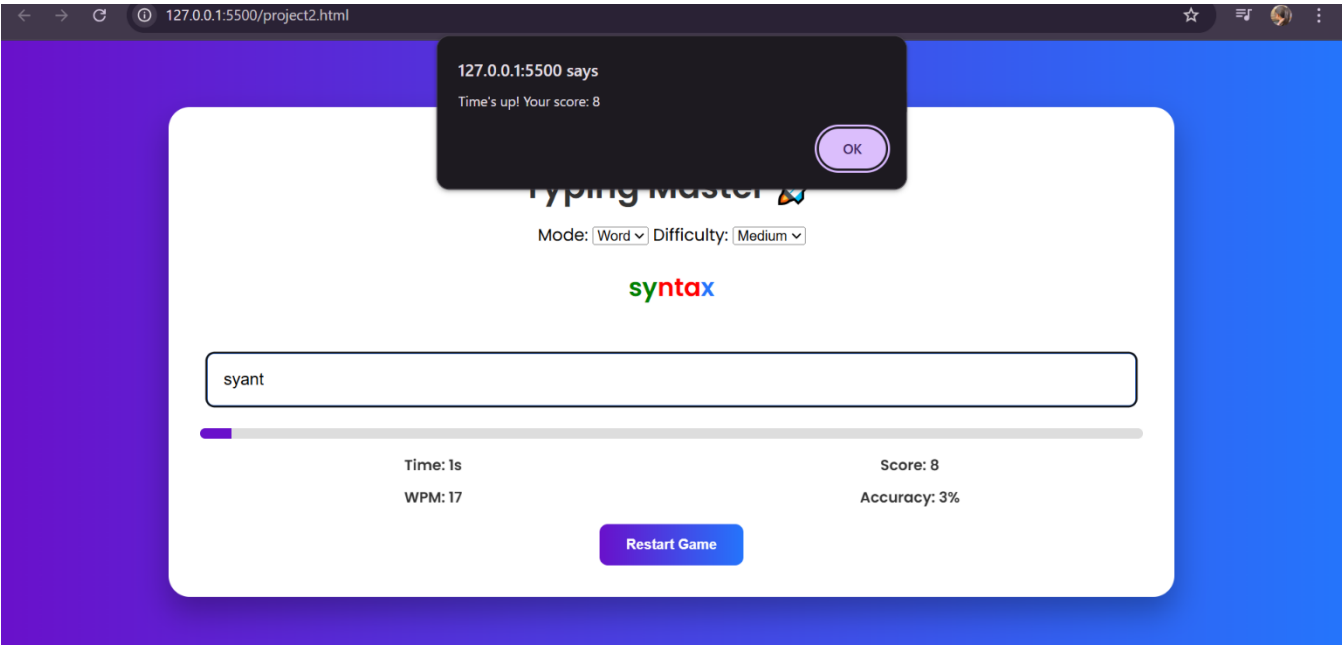
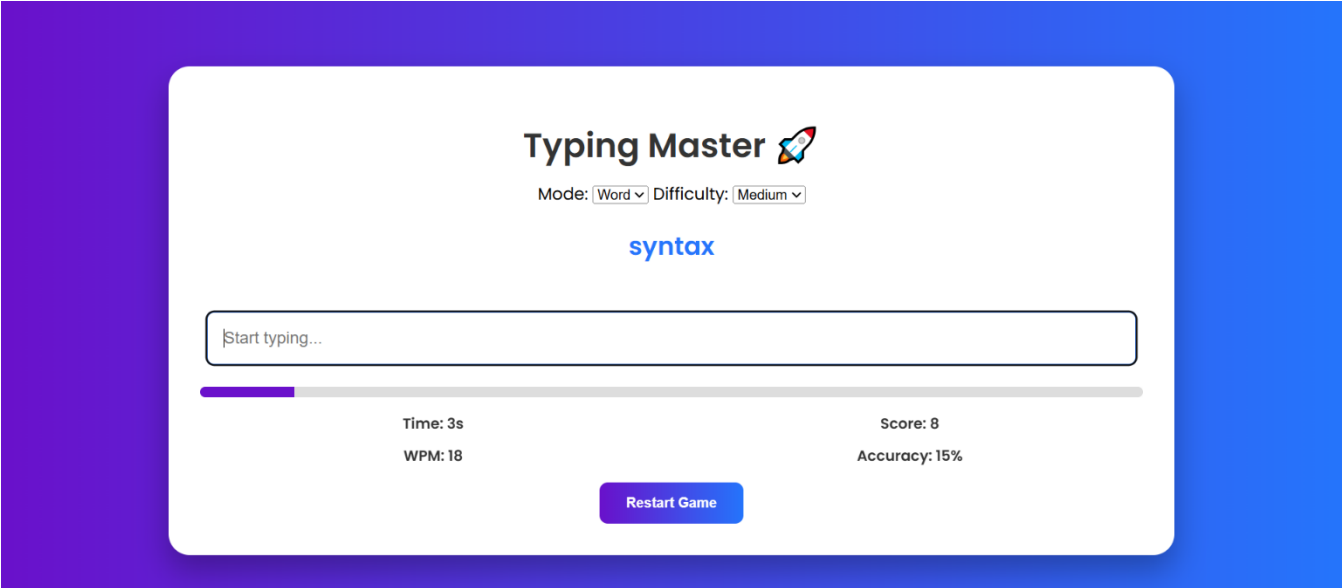
Time: 16s

WPM: 18

Score: 4

Accuracy: 19%

Restart Game



Conclusion (Extended):

The Typing Master Pro project is a practical and impactful initiative aimed at enhancing users' typing skills in an interactive and enjoyable way. By combining the core technologies of HTML, CSS, and JavaScript, the application offers a smooth and responsive interface that keeps users engaged while they practice and test their typing proficiency.

This project demonstrates how simple web technologies can come together to build an effective learning tool. The HTML structures the content neatly, while CSS adds visual appeal through creative styling, gradients, flexbox layouts, and responsive design. Meanwhile, JavaScript plays a critical role in providing interactivity, handling user inputs, controlling game flow, and dynamically updating results in real-time.

An important achievement of this project is its focus on user experience (UX). The design is clean, navigation is intuitive, and the game responds promptly to user actions. Features such as a prominent call-to-action button, loading effects, and live feedback contribute to an engaging user journey. Optional enhancements like animations and typing effects further enrich the experience, making the practice feel less monotonous and more motivating.

Additionally, the project is developed with good coding practices — separating concerns (HTML, CSS, JS), using clear naming conventions, and adding comments for readability. This ensures that the codebase is maintainable and can be expanded in the future, such as by adding new features like multiplayer mode, sound effects, scoreboards, or performance analytics.

In conclusion, Typing Master Pro is not just an academic exercise but a functional and user-friendly application that provides real value. It has laid the groundwork for further development and innovation. With continuous improvements, it can evolve into a comprehensive learning platform for students, professionals, and anyone looking to improve their typing speed and accuracy.