|  |  |
| --- | --- |
| Name-Alfiya Sayyed | Roll no-58 |
| Div-F | SRn-202202232 |

**To perform web development  for custom application using ReactJS.**

Code:

//App.js

// App.js

import React, { useState, useEffect } from 'react';

import './App.css';

const App = () => {

  const levels = [

    {

      name: 'Easy',

      paragraphs: [

        "A plant is one of the most important living things that develop on the earth and is made up of stems, leaves, roots, and so on.",

        "The root is the part of the plant that grows in the soil. The primary root emerges from the embryo."

      ]

    },

    {

      name: 'Moderate',

      paragraphs: [

        "Stem is the posterior part that remains above the ground and grows negatively geotropic. Internodes and nodes are found on the stem.",

        "It is the blossom of a plant. A flower is the part of a plant that produces seeds, which eventually become other flowers."

      ]

    },

    {

      name: 'Hard',

      paragraphs: [

        "An aunt is a bassoon from the right perspective. As far as we can estimate, some posit the melic myanmar to be less than kutcha.",

        "Some hempy laundries are thought of simply as orchids. A gum is a trumpet from the right perspective."

      ]

    }

  ];

  const [typingText, setTypingText] = useState([]);

  const [inpFieldValue, setInpFieldValue] = useState('');

  const maxTime = 60;

  const [timeLeft, setTimeLeft] = useState(maxTime);

  const [charIndex, setCharIndex] = useState(0);

  const [mistakes, setMistakes] = useState(0);

  const [isTyping, setIsTyping] = useState(false);

  const [WPM, setWPM] = useState(0);

  const [CPM, setCPM] = useState(0);

  const [currentLevel, setCurrentLevel] = useState(levels[0]);

  const [showResults, setShowResults] = useState(false); // State to show/hide results

  const [timeTaken, setTimeTaken] = useState(null); // State to store time taken

  const highlightCharacters = (inputValue, currentChar) => {

    if (inputValue === ' ') {

      return 'space';

    } else if (inputValue === currentChar.char) {

      return 'correct';

    } else if (inputValue !== '' && inputValue !== currentChar.char) {

      return 'wrong';

    } else {

      return 'untyped';

    }

  };

  const loadParagraph = () => {

    const randomIndex = Math.floor(Math.random() \* currentLevel.paragraphs.length);

    const paragraph = currentLevel.paragraphs[randomIndex].split('').map((char, index) => ({

      char: char,

      status: 'untyped',

      isMistake: false

    }));

    setTypingText(paragraph);

    setInpFieldValue('');

    setCharIndex(0);

    setMistakes(0);

    setIsTyping(false);

    setShowResults(false);

    setTimeTaken(null);

  };

  const handleInputChange = (event) => {

    const inputValue = event.target.value;

    setInpFieldValue(inputValue);

    if (!isTyping) {

      setIsTyping(true);

    }

    let updatedMistakes = 0;

    const updatedText = typingText.map((char, index) => {

      if (index < inputValue.length) {

        const status = highlightCharacters(inputValue[index], char);

        if (status === 'wrong') {

          updatedMistakes++;

        }

        return {

          ...char,

          status: status

        };

      } else {

        return {

          ...char,

          status: 'untyped'

        };

      }

    });

    setTypingText(updatedText);

    setMistakes(updatedMistakes);

  };

  const handleKeyDown = (event) => {

    if (event.key === 'Backspace' && charIndex > 0) {

      setCharIndex(charIndex - 1);

    }

  };

  const handleSubmit = () => {

    setShowResults(true);

    setIsTyping(false);

    setTimeTaken(maxTime - timeLeft);

  };

  useEffect(() => {

    if (timeLeft > 0 && isTyping) {

      const timer = setTimeout(() => {

        setTimeLeft(timeLeft - 1);

      }, 1000);

      return () => clearTimeout(timer);

    } else if (timeLeft === 0) {

      setIsTyping(false);

      setShowResults(true);

      setTimeTaken(maxTime);

    }

  }, [timeLeft, isTyping]);

  useEffect(() => {

    const totalCharacters = typingText.length;

    const elapsedTime = maxTime - timeLeft;

    const grossCPM = totalCharacters / (elapsedTime / 60);

    setCPM(Math.floor(grossCPM));

    setWPM(Math.floor(grossCPM / 5));

  }, [timeLeft, typingText]);

  useEffect(() => {

    loadParagraph();

  }, [currentLevel]);

  return (

    <div className="container">

      <h1>Typing Master</h1>

      <div>

        <label>Select Level: </label>

        <select value={currentLevel.name} onChange={(e) => setCurrentLevel(levels.find(level => level.name === e.target.value))}>

          {levels.map(level => (

            <option key={level.name} value={level.name}>{level.name}</option>

          ))}

        </select>

      </div>

      {showResults ? (

        <div className="results-container">

          <h1>Results</h1>

          <p>Time Taken: {timeTaken} seconds</p>

          <p>Mistakes: {mistakes}</p>

          <p>WPM: {WPM}</p>

          <p>CPM: {CPM}</p>

          <button onClick={loadParagraph} className="reset-btn">Restart</button>

        </div>

      ) : (

        <>

          <div className="typing-area">

            {typingText.map((char, index) => (

              <span key={index} className={`char ${char.status}`}>

                {char.char === ' ' ? '\u00A0' : char.char}

              </span>

            ))}

          </div>

          <div className="user-input-box">

            <span className="user-input">{inpFieldValue}</span>

          </div>

          <div className="stats">

            <p>Time Left: {timeLeft}s</p>

            <p>Mistakes: {mistakes}</p>

            <p>WPM: {WPM}</p>

            <p>CPM: {CPM}</p>

          </div>

          <input

            type="text"

            className="input-field"

            value={inpFieldValue}

            onChange={handleInputChange}

            onKeyDown={handleKeyDown}

            autoFocus

           // disabled={!isTyping}

          />

          <button onClick={handleSubmit} className="submit-btn">Submit</button>

        </>

      )}

    </div>

  );

};

export default App;

//App.css

.container {

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  margin-top: 50px;

  background: linear-gradient(135deg, #ff9999, #ffcc99, #ffef99, #ccff99, #99ff99);

  min-height: 100vh;

}

.typing-area {

  display: flex;

  flex-wrap: wrap;

  margin-bottom: 20px;

}

.typing-area .char {

  padding: 5px;

  font-size: 20px;

}

.typing-area .correct {

  color: green;

}

.typing-area .wrong {

  color: red;

}

.user-input-box {

  border: 1px solid #ccc;

  padding: 10px;

  margin-bottom: 20px;

}

.stats {

  margin-bottom: 20px;

}

.input-field {

  width: 300px;

  padding: 10px;

  margin-bottom: 20px;

}

.reset-btn {

  padding: 10px 20px;

  font-size: 16px;

  background-color: #007bff;

  color: #fff;

  border: none;

  cursor: pointer;

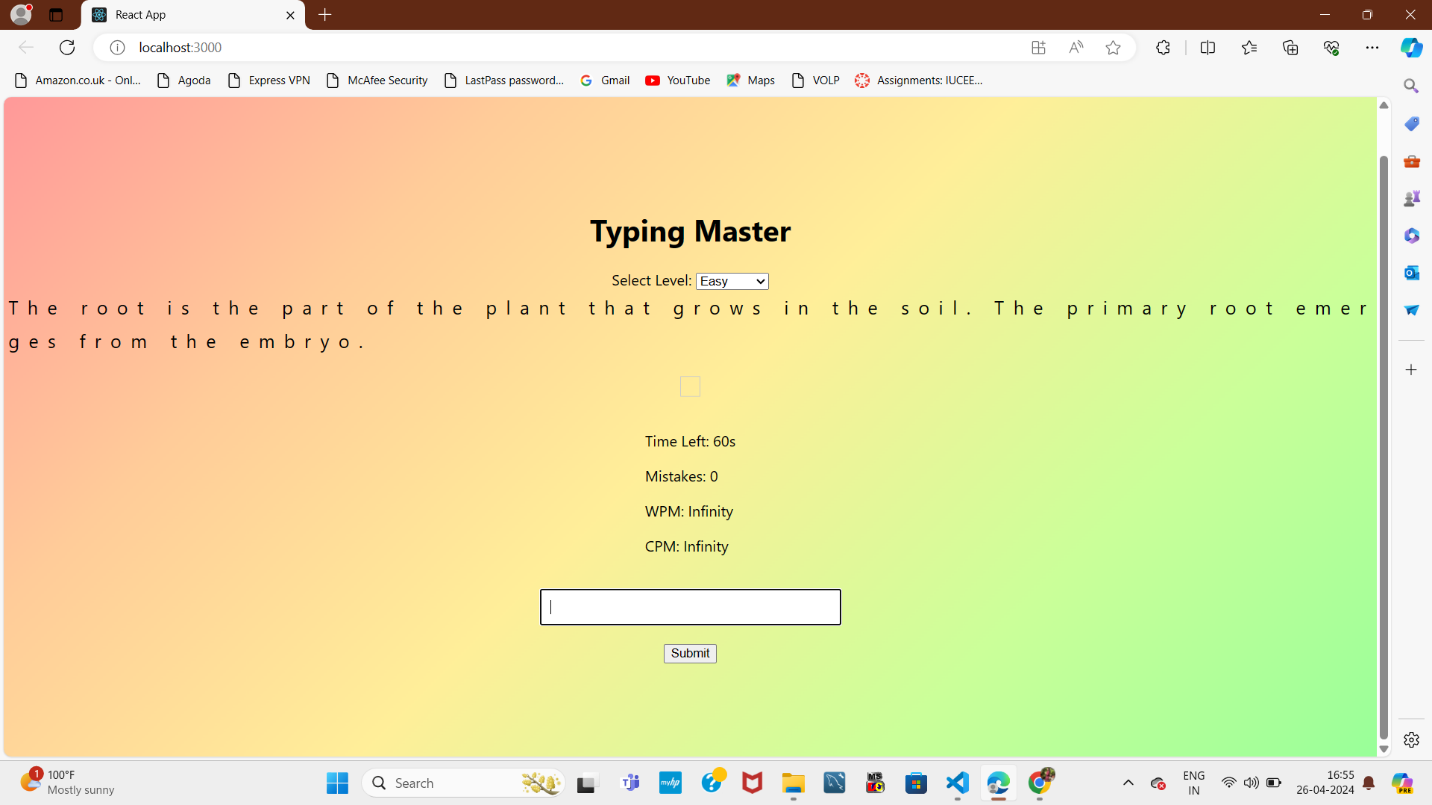
}

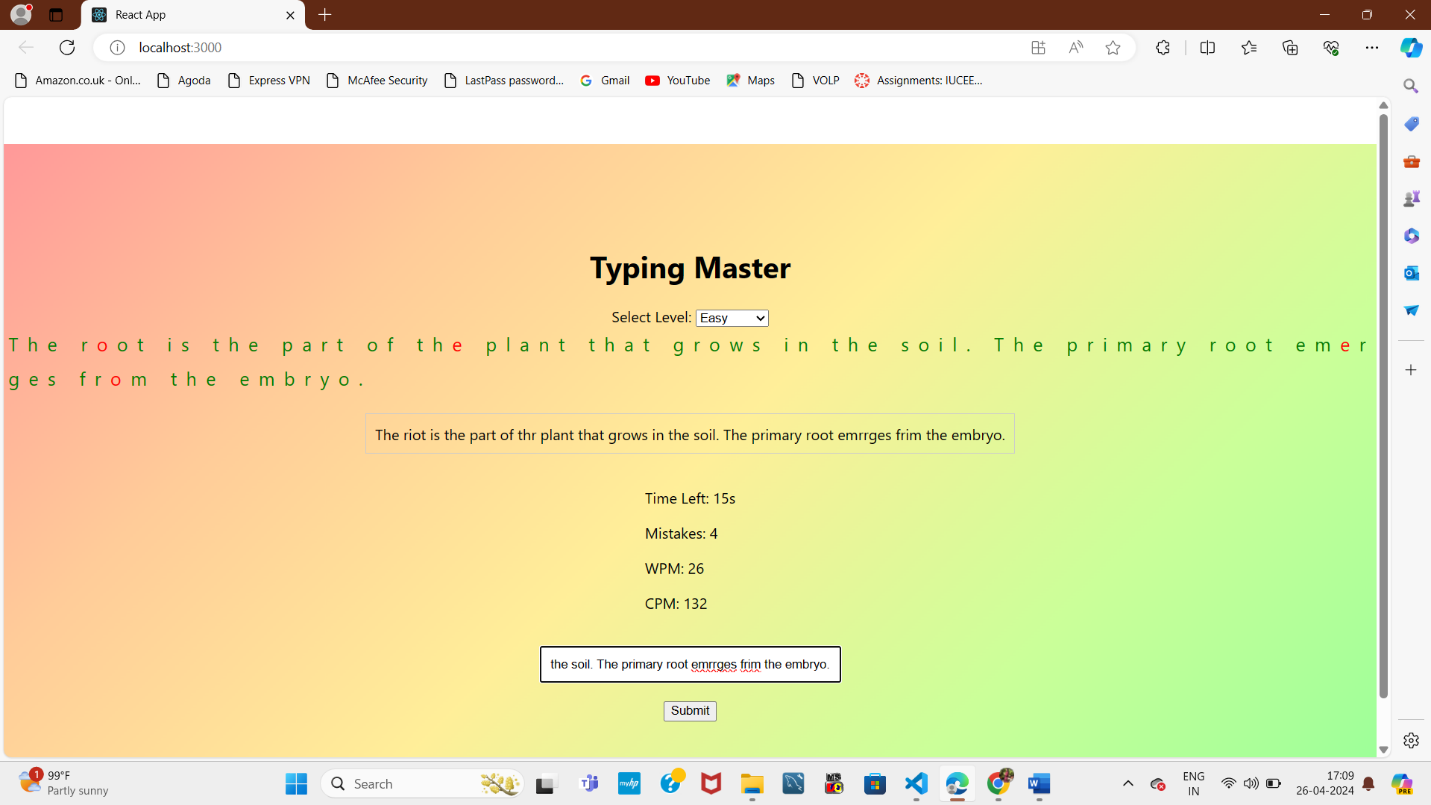
.reset-btn:hover {

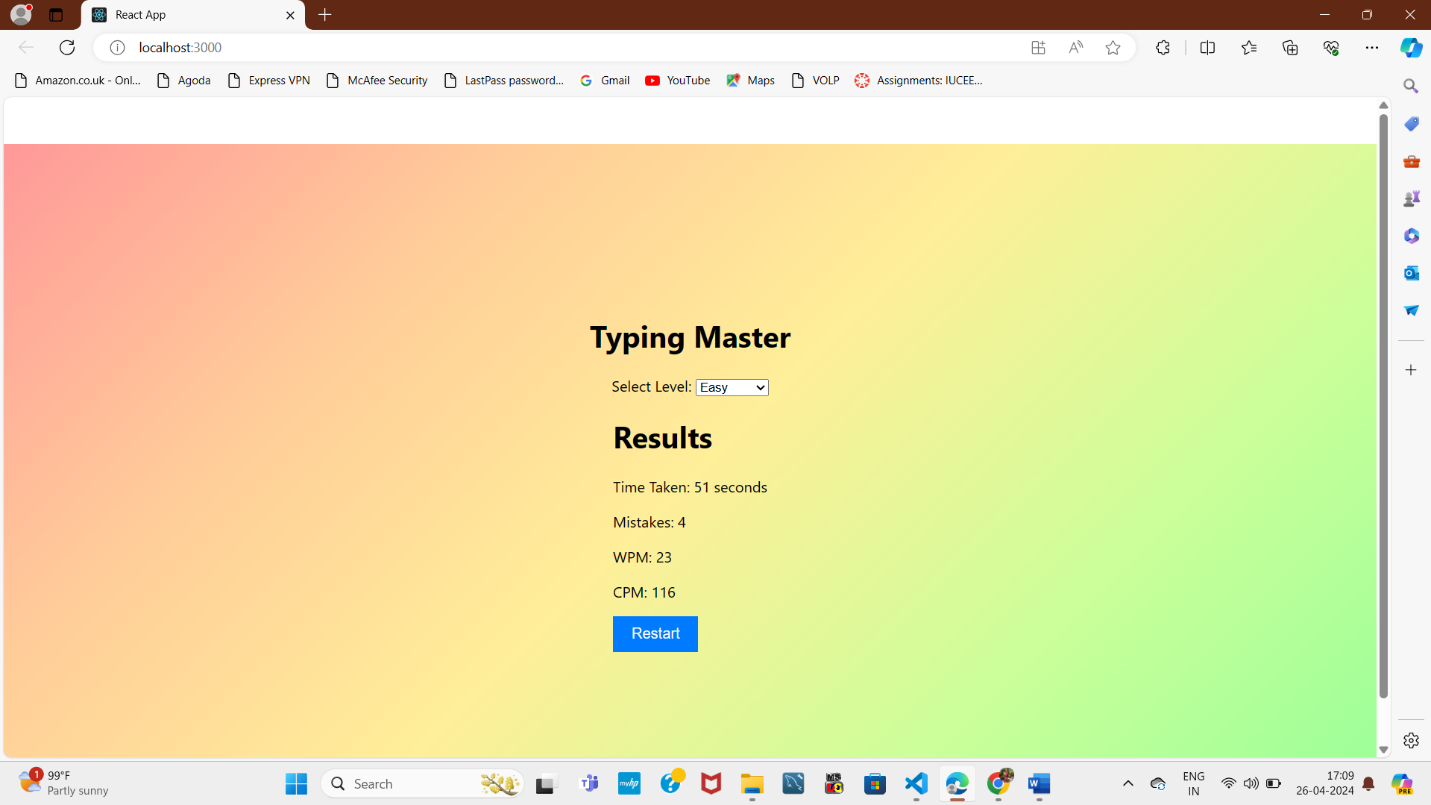
  background-color: #0056b3;

}

Output:







**To perform web development  for custom application using AngularJS.**

Code:

import { Component, Input } from '@angular/core';

@Component({

  selector: 'app-constellation-details',

  templateUrl: './constellation-details.component.html',

  styleUrls: ['./constellation-details.component.css']

})

export class ConstellationDetailsComponent {

  @Input() constellation: string;

  constellationDetails = {

    Orion: {

      name: 'Orion',

      description: 'Orion is clearly visible in the night sky from November to February. Finding Orion Belt is the easiest way to locate the Orion Constellation. Orion Belt is formed by three bright stars; Alnilam, Mintaka and Alnitak. Orion is in the southwestern sky if you are in the Northern Hemisphere or the northwestern sky if you are in the Southern Hemisphere. It is best seen between latitudes 85 and minus 75 degrees. Its right ascension is 5 hours, and its declination is 5 degrees.'

    },

    'Ursa Major': {

      name: 'Ursa Major',

      description: 'Ursa Major (also known as the Great Bear) is a constellation in the northern sky, whose associated mythology likely dates back into prehistory. Its Latin name means "greater (or larger) bear", referring to and contrasting it with nearby Ursa Minor, the lesser bear. In antiquity, it was one of the original 48 constellations listed by Ptolemy in the 2nd century AD, drawing on earlier works by Greek, Egyptian, Babylonian, and Assyrian astronomers. Today it is the third largest of the 88 modern constellations.'

    },

    Leo: {

      name: 'Leo',

      description: 'Its old astronomical symbol is  (♌︎). One of the 48 constellations described by the 2nd-century astronomer Ptolemy, Leo remains one of the 88 modern constellations today, and one of the most easily recognizable due to its many bright stars and a distinctive shape that is reminiscent of the crouching lion it depicts.'

    },

    Cassiopeia: {

      name: 'Cassiopeia',

      description: 'Cassiopeia is a constellation and asterism in the northern sky named after the vain queen Cassiopeia, mother of Andromeda, in Greek mythology, who boasted about her unrivaled beauty. Cassiopeia was one of the 48 constellations listed by the 2nd-century Greek astronomer Ptolemy, and it remains one of the 88 modern constellations today. It is easily recognizable due to its distinctive (W) shape, formed by five bright stars.'

    },

    Cygnus: {

      name: 'Cygnus',

      description: 'Cygnus is a northern constellation on the plane of the Milky Way, deriving its name from the Latinized Greek word for swan. Cygnus is one of the most recognizable constellations of the northern summer and autumn, and it features a prominent asterism known as the Northern Cross (in contrast to the Southern Cross). Cygnus was among the 48 constellations listed by the 2nd century astronomer Ptolemy, and it remains one of the 88 modern constellations.'

    }

  };

}

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  selectedConstellation: string;

  onConstellationSelect(constellation: string): void {

    this.selectedConstellation = constellation;

  }

}

Output:

