ДОДАТОК А.
Програмний код WordPool.cs

```
using System.Collections.Generic;
using UnityEngine;

public class WordPool : MonoBehaviour
{
    // завантажує json файл, після чого парсить його по стрінгах, і повертає слово, категорію та визначення
  protected List<string> words = new List<string>();
  public TextAsset jsonFile;

  [System.Serializable]
  public class WordData // об'єкт слова
  {
    public string category;
    public string word;
    public string description;
  }
  [System.Serializable]
  public class WordsData
  {
    public WordData[] words;
  }

  WordsData wordsData;
  private void Awake()
  {
    string jsonString = jsonFile.text;
    wordsData = JsonUtility.FromJson<WordsData>(jsonString);
    ConvertToLowerInvariant(wordsData.words);
  }

  private WordData RandomWord(WordData[] wordArray)
  {
    WordData randomWord = wordArray[UnityEngine.Random.Range(0, wordArray.Length)];
    string replacedString = randomWord.word.Replace(" ", "-");
    randomWord.word = replacedString;
    return randomWord;
  }

  private void ConvertToLowerInvariant(WordData[] wordArray)
  {
    foreach (WordData wordData in wordArray)
    {
      wordData.word = wordData.word.ToLower();
    }
  }

  public WordData GetWord()
  {
    return RandomWord(wordsData.words);
  }
}
```

Програмний код Keyboard.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Keyboard : MonoBehaviour
```

```
{
    [SerializeField] private WordPool wordpool;
    [SerializeField] private Text wordText;
    private string remainingWord;
    private WordPool.WordData currentWord;
    public Color redColor;
    public Color blueColor;
    [SerializeField] private AudioSource typeSound;
    [SerializeField] private AudioClip typeClip;
    private BarsManager barsManager;
    [SerializeField] private GameObject infoHolder;
    [SerializeField] private Text infoText;
    [SerializeField] private Text tryFactor;

    private Animator camAnimator;
    private int Factor = 0;


    private int lastWordLength = 0;

    private void Start()
    {
        Factor = 0;
        barsManager = FindObjectOfType<BarsManager>();
        camAnimator = Camera.main.GetComponent<Animator>();
        SetCurrentWord();
    }

    private WordPool.WordData lastWord;
    private void SetCurrentWord() // Рандомно обирає слово з загального пулу
    {
        WordPool.WordData randomWord = wordpool.GetWord();
        currentWord = null;
        while (randomWord == null || lastWord == randomWord)
        {
            randomWord = wordpool.GetWord();
        }
        currentWord = randomWord;
        lastWord = currentWord;
        SetRemainingWord(currentWord.word);
        lastWordLength = currentWord.word.Length;
    }

    private void SetRemainingWord(string newString) // Замінює слово в UI на те ж слово але без
введеної/них букв/и
    {
        wordText.text = newString;
    }

    private void Update()
    {
        CheckInput();

        if (Factor <= 0)
        {
            tryFactor.text = "";
        }
        else
        {
            tryFactor.text = "X" + Factor.ToString();
        }
    }

    private void CheckInput() // Вспоміжний метод для тестування на клавіатурі
    {
```

```
        if (Input.anyKey)
        {
            string keysPressed = Input.inputString;
            if (keysPressed.Length == 1)
            {
                EnterLetter(keysPressed);
            }
        }
    }

    public void EnterLetter(string typedLetter)
    {
        Debug.Log(Factor);
        typeSound.Play();
        if (IsCorrectLetter(typedLetter))
        {
            RemoveLetter();

            if (IsWordComplete()) // метод перевіряє що, якщо слово вже повністю прописане, то
вона дивиться категорію цього слова, і додає відповідному бару значення
            {
                Factor = (Factor + 1);
                tryFactor.GetComponent<Animator>().SetTrigger("factor");

                if (currentWord.category == "health")
                {
                    barsManager.healthBar.BarValue += 10f;
                }
                if (currentWord.category == "food")
                {
                    barsManager.foodBar.BarValue += 10f;
                }
                if (currentWord.category == "stamina")
                {
                    barsManager.staminaBar.BarValue += 10f;
                }
                if (currentWord.category == "mood")
                {
                    barsManager.moodBar.BarValue += 10f;
                }
                SetCurrentWord(); // ставить нове слово після закінчення циклу
            }
        }
        else
        {
            Debug.Log(1);
            if (Factor <= 0)
            {
                return;
            }
            Factor = (Factor - 1);
            camAnimator.SetTrigger("Wrong");
            tryFactor.GetComponent<Animator>().SetTrigger("factor");
        }
    }

    private bool IsCorrectLetter(string letter) // Повертає після введення букви наступну букву,
яку потрібно ввести у слові
    {
        string newWord = currentWord.word.Substring(currentWord.word.Length - lastWordLength);
        return newWord.StartsWith(letter);
    }

    private void RemoveLetter() // Видаляє ліву букву, після чого ділить слово на дві частини,
щоб можна було пофарбувати слово навпіл у UI
```

```
    {
        if (lastWordLength > 0)
        {
            lastWordLength -= 1;
            string leftSide = currentWord.word.Substring(0, currentWord.word.Length -
lastWordLength);
            string rightSide = currentWord.word.Substring(currentWord.word.Length -
lastWordLength);

            Color barColor = GetBarColorForCurrentWord();

            string formattedWord =
$"<color=#{ColorUtility.ToHtmlStringRGB(barColor)}>{leftSide}</color>{rightSide}";

            SetRemainingWord(formattedWord);
        }
    }

    private Color GetBarColorForCurrentWord()
    {
        if (currentWord.category == "health")
        {
            return barsManager.healthBar.GetBarColor();
        }
        else if (currentWord.category == "food")
        {
            return barsManager.foodBar.GetBarColor();
        }
        else if (currentWord.category == "stamina")
        {
            return barsManager.staminaBar.GetBarColor();
        }
        else if (currentWord.category == "mood")
        {
            return barsManager.moodBar.GetBarColor();
        }

        return Color.white;
    }

    private bool IsWordComplete()
    {
        return lastWordLength == 0;
    }
    public void SetInfoHolderActive() // ставить паузу, та виводить опис слова
    {
        barsManager.isPaused = true;
        infoHolder.SetActive(true);
        infoText.text = currentWord.description;
    }
    public void SetInfoHolderNotActive() // знімає паузу, ховає опис
    {

        barsManager.isPaused = false;
        infoHolder.SetActive(false);
    }
}
```

Програмний код ProgressBar.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


[ExecuteInEditMode]

public class ProgressBar : MonoBehaviour
{

    [Header("Title Setting")]
    public string Title;
    public Color TitleColor;
    public Font TitleFont;
    public int TitleFontSize = 10;

    [Header("Bar Setting")]
    public Color BarColor;
    public Color BarBackGroundColor;
    public Sprite BarBackGroundSprite;
    [Range(1f, 100f)]
    public int Alert = 20;
    public Color BarAlertColor;

    [Header("Sound Alert")]
    public AudioClip sound;
    public bool repeat = false;
    public float RepeatRate = 1f;

    private Image bar, barBackground;
    private float nextPlay;
    private AudioSource audiosource;
    private Text txtTitle;
    private float barValue = 1;
    public float BarValue
    {
        get { return barValue; }

        set
        {
            value = Mathf.Clamp(value, 0, 100);
            barValue = value;
            UpdateValue(barValue);
        }
    }



    private void Awake()
    {
        bar = transform.Find("Bar").GetComponent<Image>();
        barBackground = GetComponent<Image>();
        txtTitle = transform.Find("Text").GetComponent<Text>();
        barBackground = transform.Find("BarBackground").GetComponent<Image>();
        audiosource = GetComponent<AudioSource>();
    }

    private void Start()
    {
        txtTitle.text = Title;
        txtTitle.color = TitleColor;
        txtTitle.font = TitleFont;
        txtTitle.fontSize = TitleFontSize;
```

```
        bar.color = BarColor;
        barBackground.color = BarBackGroundColor;
        barBackground.sprite = BarBackGroundSprite;

        UpdateValue(barValue);


    }

    public void UpdateValue(float val)
    {
        bar.fillAmount = val / 100;
        txtTitle.text = Mathf.RoundToInt(val) + "%";

        // if (Alert >= val)
        // {
        //   bar.color = BarAlertColor;
        // }
        // else
        // {
        bar.color = BarColor;
        // }

    }


    private void Update()
    {
        if (!Application.isPlaying)
        {
            UpdateValue(50);
            txtTitle.color = TitleColor;
            txtTitle.font = TitleFont;
            txtTitle.fontSize = TitleFontSize;

            bar.color = BarColor;
            barBackground.color = BarBackGroundColor;

            barBackground.sprite = BarBackGroundSprite;
        }
        else
        {
            if (Alert >= barValue && Time.time > nextPlay)
            {
                nextPlay = Time.time + RepeatRate;
                audiosource.PlayOneShot(sound);
            }
        }
    }

    #region Api

    public Color GetBarColor()
    {
        return BarColor;
    }

    #endregion

}
```

Програмний код GameManager.cs та ButtonClickDetector.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

// Метод завантажує сцену в пам'ять проєкту, після чого визначає складність в головному меню
public class GameManager : MonoBehaviour
{
  public void StartGame()
  {
    SceneManager.LoadScene("GameScene");
  }
  public void SetDifficulty(int difficulty)
  {
    PlayerPrefs.SetInt("Difficulty", difficulty);
  }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

// Метод зчитує значення клавіші у інспекторі, після чого передає значення до менеджеру, для
подальшої обробки
public class ButtonClickDetector : MonoBehaviour
{
  private Text localText;
  private Keyboard keyboardManager;

  void Start()
  {
    keyboardManager = FindObjectOfType<Keyboard>();
    localText = GetComponentInChildren<Text>();
    Button button = GetComponent<Button>();
    button.onClick.AddListener(OnButtonClick);
  }
  void OnButtonClick()
  {
    keyboardManager.EnterLetter(localText.text);
  }
}
```

Програмний код BarsManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class BarsManager : MonoBehaviour
{
    [Header("Bars")]
    [Space]

    [SerializeField] public ProgressBar healthBar;
    [SerializeField] public ProgressBar foodBar;
    [SerializeField] public ProgressBar staminaBar;
    [SerializeField] public ProgressBar moodBar;
```

```
[SerializeField] public bool isPaused = false;

[Header("Bars Objects Colors")]
[Space]
[SerializeField] private Color InitialHealthBarColor = Color.red;
[SerializeField] private Color InitialFoodBarColor = Color.green;
[SerializeField] private Color InitialStaminaBarColor = Color.yellow;
[SerializeField] private Color InitialMoodBarColor = Color.magenta;

[Header("Objects")]
[Space]
[SerializeField] private GameObject endHolder;
[SerializeField] private GameObject playerObject;
#region Private Variable
int difficulty;
private const float InitialBarValue = 100f;
private const float UpdateInterval = 0.1f;
private Animator playerAnimator;
#endregion

private void Awake()
{
    // Знаходимо значення складності, зазделегідь визначене у меню
    difficulty = PlayerPrefs.GetInt("Difficulty", 1);

    //
    playerAnimator = playerObject.GetComponent<Animator>();
}

public void ShowEnd()
{
    endHolder.SetActive(true);
}

public void GoToMainMenu()
{
    SceneManager.LoadScene("MainScene");
}

private IEnumerator Start()
{
    InitializeBars();

    while (true)
    {
        yield return new WaitForSeconds(UpdateInterval);

        if (!isPaused)
        {
            UpdateBars();
        }
    }
}

#region Functionality
// Ініціалізація прогрес барів
private void InitializeBars()
{
    healthBar.BarValue = InitialBarValue;
    healthBar.BarColor = InitialHealthBarColor;
    foodBar.BarValue = InitialBarValue;
    foodBar.BarColor = InitialFoodBarColor;
    staminaBar.BarValue = InitialBarValue;
    staminaBar.BarColor = InitialStaminaBarColor;
    moodBar.BarValue = InitialBarValue;
```

```
        moodBar.BarColor = InitialMoodBarColor;
    }

    private void UpdateBars()
    {
        float decreaseAmount = 0.1f * difficulty;
        healthBar.BarValue -= decreaseAmount;
        foodBar.BarValue -= decreaseAmount;
        staminaBar.BarValue -= decreaseAmount;
        moodBar.BarValue -= decreaseAmount;

        if (healthBar.BarValue <= 0f || foodBar.BarValue <= 0f || staminaBar.BarValue <= 0f ||
moodBar.BarValue <= 0f)
        {
            Debug.Log("Value reached below 0!");
            ShowEnd();
        }

        CheckBarValue(healthBar, foodBar, staminaBar, moodBar);
    }

    private void CheckBarValue(ProgressBar healthBar, ProgressBar foodBar, ProgressBar
staminaBar, ProgressBar moodBar)
    {
        float hp = healthBar.BarValue;
        float food = foodBar.BarValue;
        float stamina = staminaBar.BarValue;
        float mood = moodBar.BarValue;

        if (playerObject != null)
        {
            SpriteRenderer playerSprite = playerObject.GetComponent<SpriteRenderer>();
            if (playerSprite != null)
            {
                Color targetColor = GetTargetColor(healthBar, foodBar, staminaBar, moodBar);
                Color currentColor = playerSprite.color;
                float colorChangeSpeed = 0.5f;
                float saturation = 0.5f;

                targetColor = Color.Lerp(Color.white, targetColor, saturation);
                playerSprite.color = Color.Lerp(currentColor, targetColor, colorChangeSpeed *
Time.deltaTime);
            }
        }

        //Debug.Log($" {hp} | {food} | {stamina} | {mood} ");

        if (IsAnyBarBelowThreshold(hp, food, stamina, mood))
        {
            ApplyAnims(false, true);
        }
        else if (IsAnyBarInRange(hp, food, stamina, mood))
        {
            ApplyAnims(true, false);
        }
        else
        {
            ApplyAnims(false, false);
        }
    }

    private Color GetTargetColor(ProgressBar healthBar, ProgressBar foodBar, ProgressBar
staminaBar, ProgressBar moodBar)
    {
        float hp = healthBar.BarValue;
```

```
        float food = foodBar.BarValue;
        float stamina = staminaBar.BarValue;
        float mood = moodBar.BarValue;

        float minBarVal = Mathf.Min(hp, food, stamina, mood);

        if (hp == minBarVal)
        {
            return healthBar.GetBarColor();
        }
        else if (food == minBarVal)
        {
            return foodBar.GetBarColor();
        }
        else if (stamina == minBarVal)
        {
            return staminaBar.GetBarColor();
        }
        else if (mood == minBarVal)
        {
            return moodBar.GetBarColor();
        }

        return Color.white;
    }

    private void ApplyAnims(bool isPockerface, bool isCry)
    {
        if (playerAnimator)
        {
            playerAnimator.SetBool("isPokerFace", isPockerface);
            playerAnimator.SetBool("isCry", isCry);
        }
    }
    #endregion

    #region Bools
    private bool IsAnyBarInRange(float hp, float food, float stamina, float mood)
    {
        return hp <= 75f || food <= 75f || stamina <= 75f || mood <= 75f;
    }

    private bool IsAnyBarBelowThreshold(float hp, float food, float stamina, float mood)
    {
        return hp <= 25f || food <= 25f || stamina <= 25f || mood <= 25f;
    }

    #endregion

    #region Pause
    public void PauseGame()
    {
        isPaused = true;
        Time.timeScale = 0f;
    }

    public void ResumeGame()
    {
        isPaused = false;
        Time.timeScale = 1f;
    }
    #endregion
}
```