LAMPIRAN

Listing Program

- Main *Activity*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using
UnityEngine.SceneManagement;
using TMPro;
using System;
public class Menu : MonoBehaviour
{
   // Referensi untuk menu utama,
menu tentang, menu level, dan menu
pilih karakter
   public GameObject mainMenu;
   public GameObject tentang;
   public GameObject belajar;
   public GameObject level;
   public GameObject
selectCharacterMenu; // Menu
pemilihan karakter

   private int SelectedLevelIndex =
0;
   private string selectedCharacter =
"Karakter 1"; // Karakter default
   [SerializeField] GameObject
InputNicknamePanel;
   [SerializeField] TMP_InputField
NicknameInputfield;
   [SerializeField]
TextMeshProUGUI
HomepageNickname;



  [SerializeField] List<Button>
SelectedLevelBtn = new();
  [SerializeField]
List<SelectedLevel> LevelBtns =
new();
   [SerializeField] RectTransform
SelectedLevelBorder;
   [SerializeField] Button MulaiBtn;

   // Start is called before the first
frame update
   void Start()
   {
      InitializeMenu();
   }

   private void InitializeMenu()
   {
      InitializeSavedLevel();

      // Menampilkan menu utama
dan menyembunyikan menu lainnya
saat awal
      mainMenu.SetActive(true);
      tentang.SetActive(false);
      level.SetActive(false);
      belajar.SetActive(false);
      selectCharacterMenu.SetActive(
false); // Menu karakter
disembunyikan pada awal

      if(string.IsNullOrEmpty(Game
Manager.Instance.Nickname))
      {
         InputNicknamePanel.SetActi
ve(true);
         HomepageNickname.transfor
m.parent.gameObject.SetActive(false
);
      }
      else
      {
```

```
            HomepageNickname.text =
GameManager.Instance.Nickname;
            HomepageNickname.transfor
m.parent.gameObject.SetActive(true)
;
        }

        InitializeLevelBtn();
    }

    private void InitializeSavedLevel()
    {
        if
(GameManager.Instance.SavedLevel
== null ||
GameManager.Instance.SavedLevel.
Count == 0) return;

        for (int i = 0; i <
GameManager.Instance.SavedLevel.
Count; i++)
        {
            LevelBtns[i].IsUnlocked =
GameManager.Instance.SavedLevel[
i].isUnlocked;
        }

    }

    private void InitializeLevelBtn()
    {

        for (int i = 0; i <
SelectedLevelBtn.Count; i++)
        {
            LevelBtns[i].LevelBtn =
SelectedLevelBtn[i];
            int localindex = i;
            LevelBtns[i].LevelBtn.onClic
k.AddListener(() => {
SelectLevel(localindex,
LevelBtns[localindex].LevelBtn.Get
Component<RectTransform>()); });
            LevelBtns[i].CheckLevel();
        }

        //set level start selection
        if(GameManager.Instance){
            var indexLevel =
GameManager.Instance.SelectedLev
elIndex;
            LevelBtns[indexLevel].Level
Btn.onClick.Invoke();
        }
    }
    public void OnSetNickname()
    {
        GameManager.Instance.OnSave
Nickname(NicknameInputfield.text);
        HomepageNickname.text =
GameManager.Instance.Nickname;
        HomepageNickname.transform.
parent.gameObject.SetActive(true);
        InputNicknamePanel.SetActive(
false);

    }

    // Fungsi untuk menampilkan
menu tentang
    public void tentangButton()
    {
        mainMenu.SetActive(false);
        tentang.SetActive(true);
        Debug.Log("Menu Tentang
ditampilkan.");
    }

    // Fungsi untuk menampilkan
menu level
    public void levelButton()
    {
        mainMenu.SetActive(false);
        level.SetActive(true);
        Debug.Log("Menu level
ditampilkan.");
    }

    // Fungsi untuk membuka level
sesuai dengan levelId
```

```
    private void SelectLevel(int
levelId, RectTransform btnposition)
    {
        GameManager.Instance.OnSele
ctLevel(levelId);
        if
(!SelectedLevelBorder.gameObject.a
ctiveInHierarchy)
        {
            SelectedLevelBorder.gameO
bject.SetActive(true);
        }

        MulaiBtn.interactable = true;

        SelectedLevelBorder.anchoredP
osition =
btnposition.anchoredPosition;
    }

    public void OnLoadLevel()
    {
        SceneManager.LoadScene(1);
    }

    // Fungsi untuk memeriksa apakah
scene tersedia dalam Build Settings
    private bool SceneExists(string
sceneName)
    {
        // Mengambil semua nama
scene yang ada di Build Settings
        for (int i = 0; i <
SceneManager.sceneCountInBuildSe
ttings; i++)
        {
            string scenePath =
SceneUtility.GetScenePathByBuildI
ndex(i);
            string sceneNameInBuild =
System.IO.Path.GetFileNameWithou
tExtension(scenePath);

            if (sceneNameInBuild ==
sceneName)
            {
                return true;
            }
        }
        return false;
    }

    // Fungsi untuk kembali ke menu
utama
    public void BackButton()
    {
        mainMenu.SetActive(true);
        tentang.SetActive(false);
        level.SetActive(false);
        selectCharacterMenu.SetActive(
false); // Kembali ke menu utama
dan sembunyikan menu karakter
        belajar.SetActive(false);
        Debug.Log("Kembali ke menu
utama.");
    }

    // Fungsi untuk keluar dari aplikasi
    public void QuitButton()
    {
        Debug.Log("Tombol Keluar
Telah Ditekan!...");
        // Keluar dari aplikasi
        Application.Quit();
    }

    // Fungsi untuk membuka menu
pemilihan karakter
    public void
SelectCharacterButton()
    {
        mainMenu.SetActive(false);
        selectCharacterMenu.SetActive(
true); // Menampilkan menu
pemilihan karakter
        Debug.Log("Menu Pemilihan
Karakter ditampilkan.");
    }

    // Fungsi untuk memilih karakter 1
```

```
    public void SelectCharacter1()
    {
        selectedCharacter = "Karakter 1";
        Debug.Log("Karakter 1 dipilih.");
    }

    // Fungsi untuk memilih karakter 2
    public void SelectCharacter2()
    {
        selectedCharacter = "Karakter 2";
        Debug.Log("Karakter 2 dipilih.");
    }

    // Fungsi untuk memilih karakter 3
    public void SelectCharacter3()
    {
        selectedCharacter = "Karakter 3";
        Debug.Log("Karakter 3 dipilih.");
    }

    public void SelectBelajar()
    {
        mainMenu.SetActive(false);
        belajar.SetActive(true); // Menampilkan menu belajar
        Debug.Log("Menu belajar ditampilkan.");
    }
}

[Serializable]
public class SelectedLevel
{
    public Button LevelBtn;
    private GameObject LockedIcon;

    public bool IsUnlocked = true;

    public void CheckLevel()
    {
        LockedIcon = LevelBtn.transform.GetChild(0).gameObject;

        if (!IsUnlocked)
        {
            LockedIcon.SetActive(true);
            LevelBtn.enabled = false;
        }
        else
        {
            LockedIcon.SetActive(false);
            LevelBtn.enabled = true;

        }
    }

}
```

- Player *Activity*

```
using Unity.VisualScripting;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    [SerializeField]
    private float maximumSpeed;

    [SerializeField]
    private float rotationSpeed;

    [SerializeField]
    private float jumpSpeed;

    [SerializeField]
    private float jumpButtonGracePeriod;

    [SerializeField]
    private float jumpHorizontalSpeed;
```

```
    [SerializeField]
    private Transform
cameraTransform;

    [SerializeField]
    private InputSource inputSource;

    [SerializeField]
    private Animator animator;

    [SerializeField]
    GameObject _character;

    private CharacterController
characterController;
    private float ySpeed;
    private float originalStepOffset;
    private float? lastGroundedTime;
    private float?
jumpButtonPressedTime;
    private bool isJumping;
    private bool isGrounded;

    // Reference to the dust particle
system
    public ParticleSystem dust;

    // Start is called before the first
frame update

    public static PlayerMovement
instance;

    void Awake()
    {
      instance = this;
    }
    void Start()
    {
        animator =
transform.GetChild(0).GetCompone
nt<Animator>();
        characterController =
GetComponent<CharacterController
>();
        _character.SetActive(true);
       // animator.enabled = true;
        originalStepOffset =
characterController.stepOffset;
    }

    // Update is called once per frame
    void Update()
    {
        // Handle movement input
        float horizontalInput =
inputSource.Horizontal;
        float verticalInput =
inputSource.Vertical;

        Vector3 movementDirection =
new Vector3(horizontalInput, 0,
verticalInput);
        float inputMagnitude =
Mathf.Clamp01(movementDirection.
magnitude);

        if
(Input.GetKey(KeyCode.LeftShift) ||
Input.GetKey(KeyCode.RightShift))
        {
            inputMagnitude /= 2;  //
Reduce movement speed if shift key
is pressed
        }

        animator.SetFloat("InputMagnit
ude", inputMagnitude, 0.05f,
Time.deltaTime);

        // Calculate horizontal speed
        float speed = inputMagnitude *
maximumSpeed;

        // Adjust movement direction
based on camera orientation
```

```
        movementDirection =
Quaternion.AngleAxis(cameraTransf
orm.rotation.eulerAngles.y,
Vector3.up) * movementDirection;
        movementDirection.Normalize(
);

        // Apply gravity
        if
(!characterController.isGrounded)
        {
            ySpeed += Physics.gravity.y
* Time.deltaTime;
        }

        // Handle grounding and jump
        if
(characterController.isGrounded)
        {
            lastGroundedTime =
Time.time;
            characterController.stepOffse
t = originalStepOffset;
            ySpeed = -0.5f; // Slight
downward force to stick to ground

            animator.SetBool("IsGrounde
d", true);
            isGrounded = true;
            animator.SetBool("IsJumping
", false);
            isJumping = false;
            animator.SetBool("IsFalling",
false);

            // Play dust effect when
grounded and moving
            if (inputMagnitude > 0 &&
!dust.isPlaying)
            {
                dust.Play();  // Start the
dust effect if not already playing
            }
            else if (inputMagnitude == 0
&& dust.isPlaying)
            {
                dust.Stop();  // Stop the
dust effect if player is not moving
            }

            // Handle jump input
            if
(Input.GetButtonDown("Jump") ||
isJumpingByCall ||
(jumpButtonPressedTime >
Time.time -
jumpButtonGracePeriod))
            {
                print("loncat");
                ySpeed = jumpSpeed;
                animator.SetBool("IsJumpi
ng", true);
                isJumping = true;
                isJumpingByCall = false;
                jumpButtonPressedTime =
null;
                lastGroundedTime =
null;  // Reset the jump grace time
            }
        }
        else
        {
            characterController.stepOffse
t = 0;
            animator.SetBool("IsGrounde
d", false);
            isGrounded = false;

            // Handle falling animation
            if ((isJumping && ySpeed <
0) || ySpeed < -2)
            {
                animator.SetBool("IsFallin
g", true);
            }

            // Stop the dust effect if
airborne
            if (dust.isPlaying)
            {
```

```
            dust.Stop();
        }
    }

    // Apply horizontal movement and vertical velocity
    Vector3 velocity = movementDirection * speed;
    velocity.y = ySpeed; // Include vertical movement due to gravity or jump speed

    // Move the character
    if(!GameplayManager.instance.IsGamePaused)
        characterController.Move(velocity * Time.deltaTime);

    // Rotate the character to face movement direction
    if (movementDirection != Vector3.zero)
    {
        animator.SetBool("IsMoving", true);
        Quaternion toRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
        transform.rotation = Quaternion.RotateTowards(transform.rotation, toRotation, rotationSpeed * Time.deltaTime);
    }
    else
    {
        animator.SetBool("IsMoving", false);
    }
}

private bool isJumpingByCall;

public void Jumping(){
    isJumpingByCall =true;
}

private void OnAnimatorMove()
{
    // Adjust movement based on animator delta position
    if (isGrounded)
    {
        Vector3 velocity = animator.deltaPosition;
        velocity.y = ySpeed * Time.deltaTime;

        characterController.Move(velocity);
    }
}

private void OnApplicationFocus(bool focus)
{
    /* if (focus)
    {
        Cursor.lockState = CursorLockMode.Locked;
    }
    else
    {
        Cursor.lockState = CursorLockMode.None;
    }*/
}
}
```

- Quiz *Activity*

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

public class QuisManager : MonoBehaviour
{
    // UI Elements
    public Image soalImage;
```

```
    public Button[] jawabanButtons;
    public Text feedbackText;
    public Button nextButton;
    public Button prevButton;
    public Button submitButton;

    // Data soal
    public int indexSoal;
    private List<quisData> bankSoal;
    private quisData currentSoal;

    // Menyimpan jawaban yang
dipilih oleh user untuk setiap soal
    private List<string>
selectedAnswers;
    private List<List<string>>
fixedAnswers; // Menyimpan
jawaban yang sudah diacak sekali

    void OnEnable()
    {
        initializedQuis();
    }

    // Menginisialisasi kuis dengan
data soal
    public void initializedQuis()
    {
        var data =
GameplayManager.instance.mission
Data.LevelMission;
        bankSoal = new
List<quisData>();
        selectedAnswers = new
List<string>();
        fixedAnswers = new
List<List<string>>(); // Inisialisasi
list untuk jawaban yang sudah diacak

        // Ambil soal dari LevelMission
        foreach (var mission in data)
        {
            var soal = new quisData();
            soal.data = mission;
            bankSoal.Add(soal);
        }

        // Tampilkan soal pertama

        submitButton.gameObject.SetA
ctive(false);

        ShowSoal();

        nextButton.onClick.AddListene
r(NextSoal);
        prevButton.onClick.AddListene
r(PreviousSoal);
        submitButton.onClick.AddListe
ner(SubmitAnswer);

    }

    // Menampilkan soal berdasarkan
index
    public void ShowSoal()
    {
        if (indexSoal < bankSoal.Count)
        {
            currentSoal =
bankSoal[indexSoal];
            soalImage.sprite =
currentSoal.data.spriteObj;

            // Jika jawaban belum diacak,
acak dan simpan hasilnya
            if (fixedAnswers.Count <=
indexSoal)
            {
                List<string> jawaban =
GetRandomizedAnswers(currentSoal
);
                fixedAnswers.Add(jawaba
n); // Simpan jawaban yang sudah
diacak
            }

            List<string>
jawabanTersimpan =
fixedAnswers[indexSoal];
```

```
        // Menampilkan jawaban pada tombol
        for (int i = 0; i < jawabanButtons.Length; i++)
        {
            if (i < jawabanTersimpan.Count)
            {
                var index = i;
                jawabanButtons[i].GetComponentInChildren<ArabicText>().Text = "";
                jawabanButtons[i].GetComponentInChildren<Text>().text = "";

                jawabanButtons[i].GetComponentInChildren<ArabicText>().Text = jawabanTersimpan[i];
                var arabic = jawabanButtons[i].GetComponentInChildren<ArabicText>();

                jawabanButtons[i].GetComponentInChildren<Text>().text = arabic.Text;

                print(jawabanTersimpan[i]);

                // Menambahkan listener untuk tombol
                jawabanButtons[i].onClick.RemoveAllListeners();
                string answer = jawabanTersimpan[i];
                jawabanButtons[i].onClick.AddListener(() => SelectAnswer(answer, jawabanButtons[index]));

                // Cek jika ada jawaban yang sudah dipilih
                if (selectedAnswers.Count > indexSoal && selectedAnswers[indexSoal] == jawabanTersimpan[i])
                {
                    jawabanButtons[i].transform.Find("selected/on").GetComponent<Image>().enabled = true;
                }
                else
                {
                    jawabanButtons[i].transform.Find("selected/on").GetComponent<Image>().enabled = false;
                }
            }
        }
    }

    // Menampilkan tombol submit hanya pada soal terakhir
    submitButton.gameObject.SetActive(indexSoal == bankSoal.Count - 1);
  }

  // Mendapatkan jawaban acak untuk soal
  private List<string> GetRandomizedAnswers(quisData soal)
  {
    List<string> answers = new List<string>();
    string correctAnswer = soal.data.arabicTitle;

    answers.Add(correctAnswer);

    while (answers.Count < jawabanButtons.Length)
    {
      string randomAnswer = GetRandomAnswer(answers, correctAnswer);
```

```
            if
(!string.IsNullOrEmpty(randomAns
wer))
            {
                answers.Add(randomAnsw
er);
            }
            else
            {
                break;
            }
        }

        // Acak jawaban
        for (int i = 0; i < answers.Count;
i++)
        {
            int rnd = Random.Range(i,
answers.Count);
            (answers[i], answers[rnd]) =
(answers[rnd], answers[i]);
        }

        return answers;
    }



    // Fungsi untuk mendapatkan
jawaban palsu secara acak (misalnya
dari daftar bahasa lainnya)
    private string
GetRandomAnswer(List<string>
alreadyUsed, string correctAnswer)
{
    // Prioritas: cari dari bankSoal
    List<string> sources = new();
    foreach (var soal in bankSoal)
    {
        if (soal.data.arabicTitle !=
correctAnswer &&
!alreadyUsed.Contains(soal.data.arab
icTitle))
            sources.Add(soal.data.arabic
Title);
    }

    // Jika bankSoal tidak cukup,
ambil dari Library
    if (sources.Count <
jawabanButtons.Length - 1)
    {
        var libraryAnswers =
GetDistractorAnswersFromLibrary(c
orrectAnswer);
        foreach (var ans in
libraryAnswers)
        {
            if
(!alreadyUsed.Contains(ans))
                sources.Add(ans);
        }
    }

    if (sources.Count == 0) return
null;

    return sources[Random.Range(0,
sources.Count)];
}



    private List<string>
GetDistractorAnswersFromLibrary(s
tring correctAnswer)
    {
        List<string> distractors =
new();
        if (LibraryManager.get == null
|| LibraryManager.get.library ==
null) return distractors;

        foreach (var category in
LibraryManager.get.library)
        {
            foreach (var item in
category.KumpulanData)
            {
                if
(!string.IsNullOrEmpty(item.TitleAr
```

```
ab) && item.TitleArab !=
correctAnswer &&
!distractors.Contains(item.TitleArab)
)
            {
                distractors.Add(item.Titl
eArab);
            }
        }
    }

    return distractors;
  }



  // Memilih jawaban
  public void SelectAnswer(string
answer, Button checkbtn)
  {
    // Simpan jawaban yang dipilih
    if (selectedAnswers.Count >
indexSoal)
    {
      // Hilangkan efek 'selected'
pada tombol yang sebelumnya
dipilih
      for (int i = 0; i <
jawabanButtons.Length; i++)
      {
        var itemBTN =
jawabanButtons[i];
          itemBTN.transform.Find("
selected/on").GetComponent<Image
>().enabled = false;
      }

      // Aktifkan efek 'selected'
pada tombol yang dipilih sekarang
      checkbtn.transform.Find("sel
ected/on").GetComponent<Image>().
enabled = true;
      selectedAnswers[indexSoal]
= answer; // Update jawaban jika
soal sudah dipilih
    }
    else
    {
      // Aktifkan efek 'selected'
pada tombol yang dipilih
      checkbtn.transform.Find("sel
ected/on").GetComponent<Image>().
enabled = true;
      selectedAnswers.Add(answer
); // Tambah jawaban untuk soal baru
    }
  }

  // Memeriksa jawaban yang dipilih
setelah tombol submit ditekan
  public void SubmitAnswer()
  {
    int correctAnswers = 0;

    // Mengecek semua jawaban
yang dipilih
    for (int i = 0; i <
bankSoal.Count; i++)
    {
      if (selectedAnswers.Count > i
&& selectedAnswers[i] ==
bankSoal[i].data.arabicTitle)
      {
        correctAnswers++;
      }
    }

    // Menampilkan hasil
    string feedback;
    float score =
(float)correctAnswers /
bankSoal.Count * 100;

    if (score <= 50)
    {
      feedback = $"Kuis Selesai!
\nSelamat Anda menjawab
{correctAnswers} dari
{bankSoal.Count} soal dengan
benar.\n\n Nilai: {score}%\nJangan
```

```
berkecil hati, ayo terus berusaha dan belajar lebih giat lagi!";
        }
        else
        {
            feedback = $"Kuis Selesai! \nSelamat Anda menjawab {correctAnswers} dari {bankSoal.Count} soal dengan benar.\n\n Nilai: {score}%\nBagus sekali! Terus pertahankan!";
        }

        GameplayManager.instance.WinpanelSetActive(true);
        GameplayManager.instance.QuisPanel.SetActive(false);
        GameplayManager.instance.IsGamePaused = true;


        GameplayManager.instance.WinCongratsUI.text = feedback;


    }

    // Pindah ke soal berikutnya
    public void NextSoal()
    {
        if (indexSoal < bankSoal.Count - 1)
        {
            indexSoal++;
            ShowSoal();
        }
    }

    // Pindah ke soal sebelumnya
    public void PreviousSoal()
    {
        if (indexSoal > 0)
        {
            indexSoal--;
            ShowSoal();
        }
    }
}

[System.Serializable]
public class quisData
{
    public MissionsData data;
}

[System.Serializable]
public class ItemSoal
{
    public Sprite soalImage; // Gambar soal
    public string soalDescription; // Deskripsi soal
    public string title_arabic; // Jawaban yang benar dalam bahasa Arab
}
```

- Belajar *Activity*

```
using UnityEngine;

using UnityEngine.UI;

public class Belajar : MonoBehaviour
{
    public Text Title, TitleArab, TitleLatin;
    public  Image Gambar;
    public  AudioSource Suara;
    private CategoryData Data;
    private LibraryData dataTemp;

    public void initData(string category){
        Data = LibraryManager.get.library.Find(i=> i.category == category); // ambil data dari database
```

```
        if(Data != null){
            indexKey = 0;
            showData(indexKey);
        }else{
            //data tidak ada
        }
    }

    public void showData(int index){

        if(index < Data.KumpulanData.Count){
            dataTemp = Data.KumpulanData[index];

            Gambar.sprite = dataTemp.gambar;
            Title.text = dataTemp.Title;
            TitleArab.GetComponent<ArabicText>().Text = dataTemp.TitleArab;
            TitleLatin.text = dataTemp.TitleLatin;
        }
    }

    public void showSuara(){
        Suara.PlayOneShot(dataTemp.audioTitleArab, 3);
    }

    private int indexKey = 0;
    public void switchbelajar(bool check){
        //next
        if(check){
            indexKey++;
        }
        //prev
        else {
            indexKey--;
        }

        //menjaga index tidak out of range
        indexKey = Mathf.Clamp(indexKey,0,  Data.KumpulanData.Count-1);

        showData(indexKey);
    }
}
```

- Level *Activity*

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

public class LevelManager : MonoBehaviour
{
    public static LevelManager Instance;

    void Awake()
    {
        if (Instance != null && Instance != this)
        {
            Destroy(gameObject); // Hapus duplikat
            return;
        }

        Instance = this;
        DontDestroyOnLoad(gameObject); // Menjaga agar level manager tetap ada
    }


    public List<levelData> levels;
}

[System.Serializable]
public class levelData {
    public string level;
    public bool unlocked;
    public float timergame = 60; //default set timer 1 menit
```

```
    public List<MissionsData>
LevelMission;
    public List<EnemiesData>
LevelEnemies;
}

[System.Serializable]
public class MissionsData
{
    public string title;
    public string arabicTitle;
    public string latinTitle;
    public string Descript;
    public int count;
    public Sprite spriteObj;
    public AudioClip audioTitleArab;
}

[System.Serializable]
public class EnemiesData {
    public string title;
    public GameObject prefab;
    public int count;
}
```

- Item *Activity*

```
using UnityEngine;

using UnityEngine.UI;
using TMPro;

public class ItemInfo :
MonoBehaviour
{
    public TextMeshProUGUI TITLE;
    public TextMeshProUGUI
TITLE_LATIN;
    public Text TITLE_Arabic;
    public TextMeshProUGUI
Descript;
    public TextMeshProUGUI
Counting;
    public int indexCount = 0;
    public int maxCount = 0;
    public Image ICON;
    public Image ICON_Success;

    public MissionsData data;

    // Start is called once before the
first execution of Update after the
MonoBehaviour is created

    public void
initializedInfo(ItemInfo data)
    {
        if (TITLE)
            TITLE.text =
data.TITLE.text;
        if (TITLE_LATIN)
            TITLE_LATIN.text =
data.TITLE_LATIN.text;
        if (TITLE_Arabic)
            TITLE_Arabic.text =
data.TITLE_Arabic.text;

        if (TITLE_Arabic)
            Descript.text =
data.Descript.text;

        if (ICON)
            ICON.sprite =
data.ICON.sprite;
        maxCount = data.maxCount;
    }

    public void
initializedInfoMissions(MissionsDat
a data)
    {
        if (TITLE)
            TITLE.text = data.title;
        if (TITLE_LATIN)
            TITLE_LATIN.text =
data.latinTitle;

        if (TITLE_Arabic)
        {
```

```
            TITLE_Arabic.GetCompone
nt<ArabicText>().Text =
data.arabicTitle;
            var arabicText =
TITLE_Arabic.GetComponent<Arab
icText>();
            TITLE_Arabic.text =
arabicText.Text;
        }

        if (Descript)
            Descript.text = data.Descript;

        if (ICON)
            ICON.sprite = data.spriteObj;
        maxCount = data.count;

        if (Counting)
            Counting.text = "0/" +
data.count;
    }

}
```

- Menu *Activity*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using
UnityEngine.SceneManagement;
using TMPro;
using System;
public class Menu : MonoBehaviour
{
    // Referensi untuk menu utama,
menu tentang, menu level, dan menu
pilih karakter
    public GameObject mainMenu;
    public GameObject tentang;
    public GameObject belajar;
    public GameObject level;
    public GameObject
selectCharacterMenu; // Menu
pemilihan karakter

    private int SelectedLevelIndex =
0;
    private string selectedCharacter =
"Karakter 1"; // Karakter default
    [SerializeField] GameObject
InputNicknamePanel;
    [SerializeField] TMP_InputField
NicknameInputfield;
    [SerializeField]
TextMeshProUGUI
HomepageNickname;
    [SerializeField] List<Button>
SelectedLevelBtn = new();
    [SerializeField]
List<SelectedLevel> LevelBtns =
new();
    [SerializeField] RectTransform
SelectedLevelBorder;
    [SerializeField] Button MulaiBtn;

    // Start is called before the first
frame update
    void Start()
    {
        InitializeMenu();
    }

    private void InitializeMenu()
    {
        InitializeSavedLevel();

        // Menampilkan menu utama
dan menyembunyikan menu lainnya
saat awal
        mainMenu.SetActive(true);
        tentang.SetActive(false);
        level.SetActive(false);
        belajar.SetActive(false);
        selectCharacterMenu.SetActive(
false); // Menu karakter
disembunyikan pada awal
```

```
        if(string.IsNullOrEmpty(GameManager.Instance.Nickname))
        {
            InputNicknamePanel.SetActive(true);
            HomepageNickname.transform.parent.gameObject.SetActive(false);
        }
        else
        {
            HomepageNickname.text = GameManager.Instance.Nickname;
            HomepageNickname.transform.parent.gameObject.SetActive(true);
        }

        InitializeLevelBtn();
    }

    private void InitializeSavedLevel()
    {
        if (GameManager.Instance.SavedLevel == null || GameManager.Instance.SavedLevel.Count == 0) return;

        for (int i = 0; i < GameManager.Instance.SavedLevel.Count; i++)
        {
            LevelBtns[i].IsUnlocked = GameManager.Instance.SavedLevel[i].isUnlocked;
        }

    }

    private void InitializeLevelBtn()
    {
        for (int i = 0; i < SelectedLevelBtn.Count; i++)
        {
            LevelBtns[i].LevelBtn = SelectedLevelBtn[i];
            int localindex = i;
            LevelBtns[i].LevelBtn.onClick.AddListener(() => { SelectLevel(localindex, LevelBtns[localindex].LevelBtn.GetComponent<RectTransform>()); });
            LevelBtns[i].CheckLevel();
        }

        //set level start selection
        if(GameManager.Instance){
            var indexLevel = GameManager.Instance.SelectedLevelIndex;
            LevelBtns[indexLevel].LevelBtn.onClick.Invoke();
        }
    }
    public void OnSetNickname()
    {
        GameManager.Instance.OnSaveNickname(NicknameInputfield.text);
        HomepageNickname.text = GameManager.Instance.Nickname;
        HomepageNickname.transform.parent.gameObject.SetActive(true);
        InputNicknamePanel.SetActive(false);

    }

    // Fungsi untuk menampilkan menu tentang
    public void tentangButton()
    {
        mainMenu.SetActive(false);
        tentang.SetActive(true);
        Debug.Log("Menu Tentang ditampilkan.");
    }
```

```
    // Fungsi untuk menampilkan menu level
    public void levelButton()
    {
        mainMenu.SetActive(false);
        level.SetActive(true);
        Debug.Log("Menu level ditampilkan.");
    }

    // Fungsi untuk membuka level sesuai dengan levelId
    private void SelectLevel(int levelId, RectTransform btnposition)
    {
        GameManager.Instance.OnSelectLevel(levelId);
        if (!SelectedLevelBorder.gameObject.activeInHierarchy)
        {
            SelectedLevelBorder.gameObject.SetActive(true);
        }

        MulaiBtn.interactable = true;

        SelectedLevelBorder.anchoredPosition = btnposition.anchoredPosition;
    }

    public void OnLoadLevel()
    {
        SceneManager.LoadScene(1);
    }

    // Fungsi untuk memeriksa apakah scene tersedia dalam Build Settings
    private bool SceneExists(string sceneName)
    {
        // Mengambil semua nama scene yang ada di Build Settings
        for (int i = 0; i < SceneManager.sceneCountInBuildSettings; i++)
        {
            string scenePath = SceneUtility.GetScenePathByBuildIndex(i);
            string sceneNameInBuild = System.IO.Path.GetFileNameWithoutExtension(scenePath);

            if (sceneNameInBuild == sceneName)
            {
                return true;
            }
        }
        return false;
    }

    // Fungsi untuk kembali ke menu utama
    public void BackButton()
    {
        mainMenu.SetActive(true);
        tentang.SetActive(false);
        level.SetActive(false);
        selectCharacterMenu.SetActive(false); // Kembali ke menu utama dan sembunyikan menu karakter
        belajar.SetActive(false);
        Debug.Log("Kembali ke menu utama.");
    }

    // Fungsi untuk keluar dari aplikasi
    public void QuitButton()
    {
        Debug.Log("Tombol Keluar Telah Ditekan!...");
        // Keluar dari aplikasi
        Application.Quit();
    }
```

```
    // Fungsi untuk membuka menu pemilihan karakter
    public void SelectCharacterButton()
    {
        mainMenu.SetActive(false);
        selectCharacterMenu.SetActive(true); // Menampilkan menu pemilihan karakter
        Debug.Log("Menu Pemilihan Karakter ditampilkan.");
    }

    // Fungsi untuk memilih karakter 1
    public void SelectCharacter1()
    {
        selectedCharacter = "Karakter 1";
        Debug.Log("Karakter 1 dipilih.");
    }

    // Fungsi untuk memilih karakter 2
    public void SelectCharacter2()
    {
        selectedCharacter = "Karakter 2";
        Debug.Log("Karakter 2 dipilih.");
    }

    // Fungsi untuk memilih karakter 3
    public void SelectCharacter3()
    {
        selectedCharacter = "Karakter 3";
        Debug.Log("Karakter 3 dipilih.");
    }

    public void SelectBelajar()
    {
        mainMenu.SetActive(false);
        belajar.SetActive(true); // Menampilkan menu belajar
        Debug.Log("Menu belajar ditampilkan.");
    }
}

[Serializable]
public class SelectedLevel
{
    public Button LevelBtn;
    private GameObject LockedIcon;

    public bool IsUnlocked = true;

    public void CheckLevel()
    {
        LockedIcon = LevelBtn.transform.GetChild(0).gameObject;

        if (!IsUnlocked)
        {
            LockedIcon.SetActive(true);
            LevelBtn.enabled = false;
        }
        else
        {
            LockedIcon.SetActive(false);
            LevelBtn.enabled = true;

        }
    }

}
```

- Music *Activity*

```
using UnityEngine;

public class BackgroundMusic : MonoBehaviour
{
    public AudioClip musicClip;  // AudioClip untuk musik latar
    private AudioSource audioSource;  // AudioSource untuk memutar musik
```

```
    void Start()
    {
        // Ambil komponen
AudioSource pada objek ini
        audioSource =
GetComponent<AudioSource>();

        if (audioSource == null)
        {
            // Jika tidak ada
AudioSource, buat komponen baru
            audioSource =
gameObject.AddComponent<Audio
Source>();
        }

        // Setel musik latar jika belum
ada
        if (musicClip != null)
        {
            audioSource.clip =
musicClip;
            audioSource.loop = true;  //
Memastikan musik berulang
            audioSource.Play();  //
Memulai pemutaran musik
        }
        else
        {
            Debug.LogWarning("AudioC
lip tidak diset pada
BackgroundMusic.");
        }
    }
}
```

- Time *Activity*

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

public class TimeController :
MonoBehaviour
{
    [SerializeField]
    private float timeMultiplier;

    [SerializeField]
    private float startHour;

    [SerializeField]
    private TextMeshProUGUI
timeText;

    [SerializeField]
    private Light sunLight;

    [SerializeField]
    private float sunriseHour;

    [SerializeField]
    private float sunsetHour;

    [SerializeField]
    private Color dayAmbientLight;

    [SerializeField]
    private Color nightAmbientLight;

    [SerializeField]
    private AnimationCurve
lightChangeCurve;

    [SerializeField]
    private float
maxSunLightIntensity;

    [SerializeField]
    private Light moonLight;

    [SerializeField]
    private float
maxMoonLightIntensity;

    private DateTime currentTime;

    private TimeSpan sunriseTime;
```

```
    private TimeSpan sunsetTime;

    // Start is called before the first frame update
    void Start()
    {
        currentTime = DateTime.Now.Date + TimeSpan.FromHours(startHour);

        sunriseTime = TimeSpan.FromHours(sunriseHour);
        sunsetTime = TimeSpan.FromHours(sunsetHour);
    }

    // Update is called once per frame
    void Update()
    {
        UpdateTimeOfDay();
        RotateSun();
        UpdateLightSettings();
    }

    private void UpdateTimeOfDay()
    {
        currentTime = currentTime.AddSeconds(Time.deltaTime * timeMultiplier);

        if (timeText != null)
        {
            timeText.text = currentTime.ToString("HH:mm");
        }
    }

    private void RotateSun()
    {
        float sunLightRotation;

        if (currentTime.TimeOfDay > sunriseTime && currentTime.TimeOfDay < sunsetTime)
        {
            TimeSpan sunriseToSunsetDuration = CalculateTimeDifference(sunriseTime, sunsetTime);
            TimeSpan timeSinceSunrise = CalculateTimeDifference(sunriseTime, currentTime.TimeOfDay);

            double percentage = timeSinceSunrise.TotalMinutes / sunriseToSunsetDuration.TotalMinutes;

            sunLightRotation = Mathf.Lerp(0, 180, (float)percentage);
        }
        else
        {
            TimeSpan sunsetToSunriseDuration = CalculateTimeDifference(sunsetTime, sunriseTime);
            TimeSpan timeSinceSunset = CalculateTimeDifference(sunsetTime, currentTime.TimeOfDay);

            double percentage = timeSinceSunset.TotalMinutes / sunsetToSunriseDuration.TotalMinutes;

            sunLightRotation = Mathf.Lerp(180, 360, (float)percentage);
        }

        sunLight.transform.rotation = Quaternion.AngleAxis(sunLightRotation, Vector3.right);
    }

    private void UpdateLightSettings()
```

```
    {
        float dotProduct =
Vector3.Dot(sunLight.transform.for
ward, Vector3.down);
        sunLight.intensity =
Mathf.Lerp(0,
maxSunLightIntensity,
lightChangeCurve.Evaluate(dotProd
uct));
        moonLight.intensity =
Mathf.Lerp(maxMoonLightIntensity,
0,
lightChangeCurve.Evaluate(dotProd
uct));
        RenderSettings.ambientLight =
Color.Lerp(nightAmbientLight,
dayAmbientLight,
lightChangeCurve.Evaluate(dotProd
uct));
    }

    private TimeSpan
CalculateTimeDifference(TimeSpan
fromTime, TimeSpan toTime)
    {
        TimeSpan difference = toTime -
fromTime;

        if (difference.TotalSeconds < 0)
        {
            difference +=
TimeSpan.FromHours(24);
        }

        return difference;
    }
}
```

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

PROPOSAL

Mahasiswa : Akmal	**Pembimbing I** : Wahyuddin, S.Kom., M.Kom.
NIM : 220280097	**Pembimbing II** : Marlina, S.Kom., M.Kom.
Judul Skripsi : Aplikasi Belajar Mufrodat Bahasa Arab Berbasis Game 3D Dengan Menggunakan Software Unity	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 - Tabel Use Case - Tabel flowchar		Konsultasi 1 - Tambahkan Penjelasan/Narasi Pd Desain Sistem. - Lengkapi Kerangka pikir	26/4/24
Konsultasi 2 - Teori yg digunakan masukkan di BAB II		Konsultasi 2	
Konsultasi 3 ACC		Konsultasi 3 - Perbaiki: Deskripsi Desain Sistem	29/4/24
Konsultasi 4		Konsultasi 4 ACC Ujian Proposal	
Konsultasi 5		Konsultasi 5	

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 6		Konsultasi 6	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 27 Januari 2024

Mengetahui
Ketua Program Studi

Marlina, S.Kom.,M.Kom.
NBM. 1162 680

Mahasiswa

Akmal
NIM. 220280097

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : Akmal	**Pembimbing I** : Wahyuddin, S.Kom., M.Kom.
NIM : 220280097	**Pembimbing II** : Marlina, S.Kom., M.Kom.
Judul Skripsi : Aplikasi Belajar Mufrodat Bahasa Arab Berbasis Game 3D Sebagai Media Pembelajaran	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 1. Backsound musik islami 2. karakter muslim dan muslimah 3. Menu Belajar ditambah		Konsultasi 1 - Perbaiki Daftar Isi - Perbaiki: Daftar Pustaka	
Konsultasi 2 - Potonisi pendahuluan - Daftar pustaka di tambahkan.		Konsultasi 2	
Konsultasi 3 ACC		Konsultasi 3	
Konsultasi 4 ACC ujian tutup		Konsultasi 4 Acc Ujian Hasil	
Konsultasi 5		Konsultasi 5 Perbaiki: Revisi dari Penguji pd saat Hasil	

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 6		Konsultasi 6 Bahasa Asing di Cetak Miring	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8 Acc Ujian Tutup.	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 30 April 2025

Mengetahui
Ketua Program Studi

Mughaffir Yunus, S.T., M.T.
NBM. 1252 809

Mahasiswa

Akmal
NIM. 220280097

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik