ThirdPersonController.cs (Inbuilt)

CollectingSpecs.cs

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

public class CollectingSpecs : MonoBehaviour
{
    public int specs;

    public void OnTriggerEnter(Collider Col)
    {
        if (Col.gameObject.tag == "Specs")
        {
            Debug.Log("Specs Collected!");
            specs += 1;
            SceneManager.LoadScene("GameScene");

            Destroy(Col.gameObject);
        }
    }
  }
```

GameBehvaior.cs

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

public class GameBehaviour : MonoBehaviour
{
    public static GameBehaviour Instance;

    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            Destroy(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
    }

    public void sceneToMoveTo()
    {
        SceneManager.LoadScene("GameScene");
    }
}
```

Quit.cs

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


public class Quit : MonoBehaviour
{
         public void QuitGame()
    {
        // Quit the application
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit();
#endif
    }

}
```

StartGame.cs

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

public class StartGame : MonoBehaviour
{

    public void onscene()
    {
        SceneManager.LoadScene("Playground");
    }

}
```

ShuffleList.cs

```
using System.Collections.Generic;

public abstract class ShuffleList
{
    public static List<E> KeepOriginalOrder<E>(List<E> inputList)
    {
        List<E> originalList = new List<E>();
        originalList.AddRange(inputList);

        return originalList; // Return the original list without shuffling
    }
}
```

QuizManager.cs

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

public class QuizManager : MonoBehaviour
{

#pragma warning disable 649
    //ref to the QuizGameUI script
    [SerializeField] private QuizGameUI quizGameUI;
    //ref to the scriptableobject file
    [SerializeField] private List<QuizDataScriptable> quizDataList;
    [SerializeField] private float timeInSeconds;
#pragma warning restore 649


    private string currentCategory = "";
    public int correctAnswerCount = 0;
    //questions data
    private List<Question> questions;
    //current question data
    private Question selectedQuetion = new Question();
    private int gameScore;
    private int lifesRemaining;
    private float currentTime;
    private QuizDataScriptable dataScriptable;

    private GameStatus gameStatus = GameStatus.NEXT;

    public GameStatus GameStatus { get { return gameStatus; } }
public List<QuizDataScriptable> QuizData { get => quizDataList; }

public Text ScoreText;







public void StartGame(int categoryIndex, string category)
{
    currentCategory = category;
    correctAnswerCount = 0;
    gameScore = 0;
    lifesRemaining = 3;
    currentTime =900f;
    //set the questions data
    questions = new List<Question>();
    dataScriptable = quizDataList[categoryIndex];
    questions.AddRange(dataScriptable.questions);
    //select the question
    SelectQuestion();
    gameStatus = GameStatus.PLAYING;
}

/// <summary>
/// Method used to randomly select the question form questions data
/// </summary>
private void SelectQuestion()
{
    //get the random number
    int val = UnityEngine.Random.Range(0, questions.Count);
    //set the selectedQuetion
    selectedQuetion = questions[val];
    //send the question to quizGameUI
    quizGameUI.SetQuestion(selectedQuetion);

    questions.RemoveAt(val);
}

private void Update()
{
    PlayerPrefs.SetInt("QuizToPlayground", 1);
    if (gameStatus == GameStatus.PLAYING)
    {
        currentTime -= Time.deltaTime;
        SetTime(currentTime);
    }
}

void SetTime(float value)
{
        TimeSpan time = TimeSpan.FromSeconds(currentTime);                    //set the time
value
        quizGameUI.TimerText.text = time.ToString("mm':'ss");   //convert time to Time format

        if (currentTime <= 0)
        {
            //Game Over
            GameEnd();
        }
    }

    /// <summary>
    /// Method called to check the answer is correct or not
    /// </summary>
    /// <param name="selectedOption">answer string</param>
    /// <returns></returns>
    public bool Answer(string selectedOption)
    {
        //set default to false
        bool correct = false;
        //if selected answer is similar to the correctAns
        if (selectedQuetion.correctAns == selectedOption)
        {
            //Yes, Ans is correct
            correctAnswerCount += 50;
            correct = true;
            gameScore += 50;
            quizGameUI.ScoreText.text = "Score:" + gameScore;
        }
        else
        {
            //No, Ans is wrong
            //Reduce Life
            lifesRemaining--;
            quizGameUI.ReduceLife(lifesRemaining);

            if (lifesRemaining == 0)
            {
                GameEnd();
            }
        }

        if (gameStatus == GameStatus.PLAYING)
        {
            if (questions.Count > 0)
            {
                //call SelectQuestion method again after 1s
                Invoke("SelectQuestion", 0.4f);
            }
            else
            {
                GameEnd();
            }
        }
        //return the value of correct bool
        return correct;
    }



    private void GameEnd()
    {
        gameStatus = GameStatus.NEXT;
        quizGameUI.GameOverPanel.SetActive(true);

        // Save the score
        PlayerPrefs.SetInt(currentCategory, correctAnswerCount);
        //Text ScoreText = quizGameUI.GetComponent<Text>();
        //ScoreText.text = "Score: " + correctAnswerCount;

        ScoreText.gameObject.SetActive(true);
        // Load the main menu scene after a delay
        StartCoroutine(LoadMainMenu());
    }

    private IEnumerator LoadMainMenu()
    {
        yield return new WaitForSeconds(3f); // Wait for 3 seconds before loading the main
menu
        SceneManager.LoadScene("Playground"); // Replace with your main menu scene
name
    }
}

//Datastructure for storeing the quetions data
[System.Serializable]
public class Question
{
    public string questionInfo;         //question text
    public QuestionType questionType;   //type
    public Sprite questionImage;        //image for Image Type
    public AudioClip audioClip;         //audio for audio type
    public UnityEngine.Video.VideoClip videoClip;   //video for video type
    public List<string> options;        //options to select
    public string correctAns;           //correct option
}

[System.Serializable]
public enum QuestionType
{
    TEXT,
    IMAGE,
    AUDIO,
    VIDEO
}

[SerializeField]
public enum GameStatus
```

```
{
   PLAYING,
   NEXT
}
```

QuizGameUI.cs

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


public class QuizGameUI : MonoBehaviour
{
#pragma warning disable 649
   [SerializeField] private QuizManager quizManager;            //ref to the QuizManager
script
   [SerializeField] private CategoryBtnScript categoryBtnPrefab;
   [SerializeField] private GameObject scrollHolder;
   [SerializeField] private Text scoreText, timerText;
   [SerializeField] private List<Image> lifeImageList;
   [SerializeField] private GameObject gameOverPanel, mainMenu, gamePanel;
   [SerializeField] private Color correctCol, wrongCol, normalCol; //color of buttons
   [SerializeField] private Image questionImg;                 //image component to show
image
   [SerializeField] private UnityEngine.Video.VideoPlayer questionVideo;   //to show video
   [SerializeField] private AudioSource questionAudio;           //audio source for audio clip
   [SerializeField] private Text questionInfoText;               //text to show question
   [SerializeField] private List<Button> options;                //options button reference

#pragma warning restore 649

   private float audioLength;          //store audio length
   private Question question;          //store current question data
   private bool answered = false;      //bool to keep track if answered or not

   public Text TimerText { get => timerText; }                   //getter
   public Text ScoreText { get => scoreText; }                   //getter
   public GameObject GameOverPanel { get => gameOverPanel; }                  //getter

   private void Start()
   {
      //add the listner to all the buttons
      for (int i = 0; i < options.Count; i++)
      {
         Button localBtn = options[i];
         localBtn.onClick.AddListener(() => OnClick(localBtn));
      }
      CategoryBtn(2, "Mix");
```

```
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
        CreateCategoryButtons();
        gamePanel.SetActive(true);
        mainMenu.SetActive(false);

    }
    /// <summary>
    /// Method which populate the question on the screen
    /// </summary>
    /// <param name="question"></param>
    public void SetQuestion(Question question)
    {
        //set the question
        this.question = question;
        //check for questionType
        switch (question.questionType)
        {
            case QuestionType.TEXT:
                questionImg.transform.parent.gameObject.SetActive(false);   //deactivate image
holder
                break;
            case QuestionType.IMAGE:
                questionImg.transform.parent.gameObject.SetActive(true);    //activate image
holder
                questionVideo.transform.gameObject.SetActive(false);        //deactivate
questionVideo
                questionImg.transform.gameObject.SetActive(true);           //activate questionImg
                questionAudio.transform.gameObject.SetActive(false);        //deactivate
questionAudio

                questionImg.sprite = question.questionImage;                //set the image sprite
                break;
            case QuestionType.AUDIO:
                questionVideo.transform.parent.gameObject.SetActive(true);  //activate image
holder
                questionVideo.transform.gameObject.SetActive(false);        //deactivate
questionVideo
                questionImg.transform.gameObject.SetActive(false);          //deactivate
questionImg
                questionAudio.transform.gameObject.SetActive(true);         //activate
questionAudio

                audioLength = question.audioClip.length;                    //set audio clip
                StartCoroutine(PlayAudio());                                //start Coroutine
                break;
            case QuestionType.VIDEO:
                questionVideo.transform.parent.gameObject.SetActive(true);  //activate image
holder
                questionVideo.transform.gameObject.SetActive(true);         //activate
questionVideo
                questionImg.transform.gameObject.SetActive(false);          //deactivate
questionImg
                questionAudio.transform.gameObject.SetActive(false);        //deactivate
questionAudio

            questionVideo.clip = question.videoClip;                //set video clip
            questionVideo.Play();                                   //play video
            break;
      }

      questionInfoText.text = question.questionInfo;                //set the question text

      //suffle the list of options
      //List<string> ansOptions = ShuffleList.ShuffleListItems<string>(question.options);
      List<string> ansOptions = ShuffleList.KeepOriginalOrder<string>(question.options);

      //assign options to respective option buttons
      for (int i = 0; i < options.Count; i++)
      {
         //set the child text
         options[i].GetComponentInChildren<Text>().text = ansOptions[i];
         options[i].name = ansOptions[i];    //set the name of button
         options[i].image.color = normalCol; //set color of button to normal
      }

      answered = false;

   }

   public void ReduceLife(int remainingLife)
   {
      lifeImageList[remainingLife].color = Color.red;
   }

   /// <summary>
   /// IEnumerator to repeate the audio after some time
   /// </summary>
   /// <returns></returns>
   IEnumerator PlayAudio()
   {
      //if questionType is audio
      if (question.questionType == QuestionType.AUDIO)
      {
         //PlayOneShot
         questionAudio.PlayOneShot(question.audioClip);
         //wait for few seconds
         yield return new WaitForSeconds(audioLength + 0.5f);
         //play again
         StartCoroutine(PlayAudio());
      }
      else //if questionType is not audio
      {
         //stop the Coroutine
         StopCoroutine(PlayAudio());
         //return null
         yield return null;
      }
```

```
    }

    /// <summary>
    /// Method assigned to the buttons
    /// </summary>
    /// <param name="btn">ref to the button object</param>
    void OnClick(Button btn)
    {
        if (quizManager.GameStatus == GameStatus.PLAYING)
        {
            //if answered is false
            if (!answered)
            {
                //set answered true
                answered = true;
                //get the bool value
                bool val = quizManager.Answer(btn.name);

                //if its true
                if (val)
                {
                    //set color to correct
                    //btn.image.color = correctCol;
                    StartCoroutine(BlinkImg(btn.image));
                }
                else
                {
                    //else set it to wrong color
                    btn.image.color = wrongCol;
                }

            }
        }
    }



    /// <summary>
    /// Method to create Category Buttons dynamically
    /// </summary>
    void CreateCategoryButtons()
    {
        //we loop through all the available catgories in our QuizManager
        for (int i = 0; i < quizManager.QuizData.Count; i++)
        {
            //Create new CategoryBtn
            CategoryBtnScript categoryBtn = Instantiate(categoryBtnPrefab,
scrollHolder.transform);
            //Set the button default values
            categoryBtn.SetButton(quizManager.QuizData[i].categoryName,
quizManager.QuizData[i].questions.Count);
            int index = i;
            //Add listner to button which calls CategoryBtn method
            categoryBtn.Btn.onClick.AddListener(() => CategoryBtn(index,
quizManager.QuizData[index].categoryName));
        }
    }

    //Method called by Category Button
    private void CategoryBtn(int index, string category)
    {
        quizManager.StartGame(index, category); //start the game
        mainMenu.SetActive(false);              //deactivate mainMenu
        gamePanel.SetActive(true);              //activate game panel
    }

    //this give blink effect [if needed use or dont use]
    IEnumerator BlinkImg(Image img)
    {
        for (int i = 0; i < 2; i++)
        {
            img.color = Color.white;
            yield return new WaitForSeconds(0.1f);
            img.color = correctCol;
            yield return new WaitForSeconds(0.1f);
        }
    }

    public void RestryButton()
    {
        SceneManager.LoadScene("Playground");

    }

    public void QuitGame()
    {
        // Quit the application when the Quit button is clicked
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit();
#endif
    }

}
```

QuizDataScriptable.cs

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

[CreateAssetMenu(fileName = "QuestionsData", menuName = "QuestionsData", order =
1)]
```

```
public class QuizDataScriptable : ScriptableObject
{
   public string categoryName;
   public List<Question> questions;
}
```

FinalScoreDisplay.cs

```
using UnityEngine;
using UnityEngine.UI;

public class FinalScoreDisplay : MonoBehaviour
{
   [SerializeField] private Text ScoreText;
   [SerializeField] private Text MessageText;
   [SerializeField] private QuizManager quizManager; // Reference to your QuizManager
script

   private void Start()
   {
      if (quizManager != null)
      {
         // Set the final score text
         int finalScore = quizManager.correctAnswerCount;
         ScoreText.text = "Final Score: " + finalScore.ToString();

         // Compare the final score and display a message
         if (finalScore >= 400)
         {
            MessageText.text = "Congratulations! Precise Eyesight!";
         }
         else if((finalScore >= 200) && (finalScore < 400 ) )
         {
            MessageText.text = "Keep practicing for better results!";
         }
         else
         {
            MessageText.text = "You have Weak Eyesight!!";
         }
      }
      else
      {
         Debug.LogWarning("QuizManager reference not set!");
      }
   }
}
```

CategoryBtn.cs

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

public class CategoryBtnScript : MonoBehaviour
{
    [SerializeField] private Text categoryTitleText;
    [SerializeField] private Text scoreText;
    [SerializeField] private Button btn;

    public Button Btn { get => btn; }

    public void SetButton(string title, int totalQuestion)
    {
        categoryTitleText.text = title;
        scoreText.text = PlayerPrefs.GetInt(title, 0) + "/" + totalQuestion; //we get the score
save for this category
    }

}
```