Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama:
1. Deva Ahmad (122140015)
2. Faris Pratama (122140021)
3. Arkan Hariz Chandra Winata Liem (122140038)

Tugas Ke: **Final Project**

Mata Kuliah: **Teknologi Multimedia (IF-40305)**

Tanggal: December 2, 2025

1 Deskripsi Proyek

Guess The Word adalah filter kuis yang dimana pengguna menebak kata dari audio. Nada dari audio tersebut akan dibuat seperti chipmunk dan audio akan menyebutkan satu kata. Kata yang disebutkan dalam Bahasa Indonesia dan/atau Bahasa inggris. Terdapat pilihan ganda yang mana pengguna akan memilih jawaban yang sesuai dengan kata yang disebutkan oleh audio. pengguna akan memilih dengan cara menunjuk atau mengarahkan jari ke jawaban yang dipilih. Kuis ini terdapat 3 kategori tebak-tebakan. Selain itu, jawaban yang benar akan mendapatkan poin dan jawaban yang salah tidak dapat poin. Poin akan ditotalkan diakhir kuis. Filter ini cocok untuk melatih pendengaran dan pengucapan terhadap suatu kosakata, sehingga menambah gudang kosakata.

2 Teknologi

Teknologi yang digunakan dalam mengerjakan Final Project ini, yaitu sebagai berikut:

- **Python**
  Bahasa pemrograman utama yang digunakan dalam pengembangan aplikasi.
- **gTTS**
  Library untuk mengubah teks menjadi suara menggunakan layanan text-to-speech Google. Output biasanya berupa file audio seperti.
- **MediaPipe**
  Library untuk computer vision dan machine learning real-time, seperti deteksi tangan, pose tubuh, wajah, gesture tracking, face mesh, dan lain-lain.
- **pydub**
  Library untuk memanipulasi audio, seperti memotong, menggabungkan, mengubah volume, format audio, konversi .mp3 ke .wav, dan lain-lain.
- **OpenCV2**
  Library untuk pengolahan gambar dan video, seperti deteksi objek, face detection, filtering, drawing, membaca/menyimpan file video, dan pengolahan piksel.
- **Librosa**
  Library untuk analisis dan pemrosesan audio dan musik, seperti menghitung spektrogram, MFCC, tempo, pitch, waveform, dan fitur audio lainnya untuk machine learning atau DSP.

- **Soundfile**
  Library untuk membaca dan menulis berbagai format file audio (WAV, FLAC, OGG, AIFF) dan bekerja bersama dengan NumPy untuk mengakses data audio dalam bentuk array.
- **NumPy**
  Library untuk komputasi numerik berbasis array. Digunakan untuk operasi matematika cepat, manipulasi data berbentuk matriks, dan banyak dipakai sebagai basis data pada pengolahan audio/gambar.
- **Pygame**
  Library untuk membuat game 2D dan multimedia. Digunakan untuk mengelola audio (memutar sound effect, musik latar), menangani input pengguna, dan rendering grafis sederhana.

3 Cara Kerja Sistem

- Mendeteksi jari tangan dengan kamera dan interaksinya terhadap komponen lain dalam sistem.
- Memutar suara berdasarkan kategori dengan sudah dinaikan pitch-nya sehingga menjadi tantangan dalam menebak.
- Menampilkan pilihan-pilihan untuk menjawab dari apa yang disebutkan oleh suara.
- Memunculkan indikasi jawaban benar atau salah dari suara yang muncul setelah menjawab, seperti "Ting" jika benar dan "Tot" jika salah.
- Menampilkan skor dari jawaban yang sudah dijawab dan bertambah jika jawaban benar kemudian menampilkan skor diakhir game.

4 Penjelasan Kode Program

Kode program pada filter Guess The Word terdiri dari 6 bagian utama, yaitu:

- File **main** sebagai program utama.
- File *config* sebagai konfigurasi awal.
- File *audio_manager* sebagai pengelola audio.
- File *audio_code* sebagai penghasil audio dari teks.
- File *audio_processing* sebagai pengolah audio (pitching).
- File *graphics_manager* sebagai pengelola grafis.
- File *game_logic* sebagai logika permainan.
- File *utils* sebagai fungsi tambahan.

Berikut adalah penjelasan dari fungsi masing-masing file:

4.1 File - Main

Secara konsep, file ini berfungsi sebagai pengendali alur program. Dimulai dari:

- Inisialisasi
  - Kamera (OpenCV)
  - Deteksi tangan/jari (MediaPipe)
  - Sistem audio (pygame)
  - Asset gambar tombol (PNG)
  - Data soal per kategori
- Menjalankan *Game State* dengan 4 state
  - STATE_HOME terdiri dari halaman awal, header dan tombol "Play"
  - STATE_CATEGORY terdiri dari pilih kategori (buah, hewan, kendaraan)
  - STATE_GAME terdiri dari proses kuis berjalan (audio dengan 4 pilihan jawaban)
  - STATE_RESULT terdiri dari tampilan skor serta tombol "Mainkan Lagi"
- Menangani
  - Posisi jari sebagai "cursor"
  - Deteksi klik tombol (berdasarkan alpha PNG)
  - Cooldown 2 detik biar tombol gak spam
  - Delay sebelum pindah soal
  - Efek tombol "naik" sedikit saat disentuh

Berikut ini adalah implementasi kode pada file **main.py**:

```python
import cv2
import mediapipe as mp
import numpy as np
import pygame  # untuk quit mixer di akhir

from src.config import (
    WINDOW_NAME,
    HEADER_BUTTON_PATH, PLAY_BUTTON_PATH,
    CAT_BUAH_PATH, CAT_HEWAN_PATH, CAT_KENDARAAN_PATH,
    PLAY_AGAIN_PATH, PILIH_KATEGORI_PATH,
    REPEAT_AUDIO_PATH,
    STATE_HOME, STATE_CATEGORY, STATE_GAME, STATE_RESULT,
    QUESTION_DELAY_MS,
)
from src.audio_manager import (
    play_click_sfx, play_correct, play_wrong,
    play_applause, play_boo,
    update_bgm_for_state, AUDIO_OK,
)
from src.graphics import (
    load_button_image, overlay_png, draw_button_with_effect,
    point_on_png_button, put_text_with_outline,
    load_all_answer_assets,
)
from src.game_logic import (
    build_questions_for_category, setup_question,
)
```

```python
from src.utils import now_ms, can_interact, set_cooldown

# ========================
# LOAD ASSET IMAGE
# ========================

header_btn = load_button_image(HEADER_BUTTON_PATH, size=(400, 100))
play_btn   = load_button_image(PLAY_BUTTON_PATH,   size=(260, 100))

CAT_BTN_SIZE = (380, 130)
cat_buah_btn      = load_button_image(CAT_BUAH_PATH,      CAT_BTN_SIZE)
cat_hewan_btn     = load_button_image(CAT_HEWAN_PATH,     CAT_BTN_SIZE)
cat_kendaraan_btn = load_button_image(CAT_KENDARAAN_PATH, CAT_BTN_SIZE)

kategori_header_btn = load_button_image(PILIH_KATEGORI_PATH, size=(400, 100))

REPEAT_BTN_SIZE = (260, 80)
repeat_audio_btn = load_button_image(REPEAT_AUDIO_PATH, REPEAT_BTN_SIZE)

PLAY_AGAIN_BTN_SIZE = (320, 110)
play_again_btn = load_button_image(PLAY_AGAIN_PATH, PLAY_AGAIN_BTN_SIZE)

ALL_ANSWER_ASSETS = load_all_answer_assets()

# ========================
# STATE & VARIABEL GAME
# ========================

state = STATE_HOME
selected_category = None

score = 0
questions = []
current_q_index = 0

current_q = None
correct_answer = None
answer_buttons = []

question_sound = None
question_start_time = 0
audio_initial_played = False
repeat_used = False
result_sound_played = False

question_answered = False
question_transition_deadline = None

print("Tekan 'q' untuk keluar.")

# ========================
# SETUP MEDIAPIPE & KAMERA
# ========================

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils

hands = mp_hands.Hands(
    static_image_mode=False,
    max_num_hands=1,
    min_detection_confidence=0.6,
    min_tracking_confidence=0.6,
```

```python
    )

    cv2.namedWindow(WINDOW_NAME, cv2.WINDOW_NORMAL)
    cv2.setWindowProperty(
        WINDOW_NAME,
        cv2.WND_PROP_FULLSCREEN,
        cv2.WINDOW_FULLSCREEN
    )

    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        print("Error: Kamera tidak dapat diakses.")
        exit()

    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

    # ========================
    # MAIN LOOP
    # ========================

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame = cv2.flip(frame, 1)
        h_frame, w_frame, _ = frame.shape
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        results = hands.process(frame_rgb)

        fingertip = None

        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(
                    frame,
                    hand_landmarks,
                    mp_hands.HAND_CONNECTIONS,
                    mp_drawing.DrawingSpec(color=(0, 255, 0), thickness=2, circle_radius=2),
                    mp_drawing.DrawingSpec(color=(0, 0, 255), thickness=2),
                )

                cx = int(hand_landmarks.landmark[8].x * w_frame)
                cy = int(hand_landmarks.landmark[8].y * h_frame)
                fingertip = (cx, cy)

                cv2.circle(frame, (cx, cy), 10, (255, 255, 0), cv2.FILLED)
                break

        header_x = (w_frame - header_btn.shape[1]) // 2
        header_y = 20

        # ---------- STATE HOME ----------
        if state == STATE_HOME:
            frame = overlay_png(frame, header_btn, header_x, header_y)

            play_x = (w_frame - play_btn.shape[1]) // 2
            play_y = int(h_frame * 0.7)
            frame = draw_button_with_effect(frame, play_btn, play_x, play_y, fingertip)

```

```python
            if fingertip and can_interact():
                if point_on_png_button(fingertip[0], fingertip[1], play_x, play_y, play_btn):
                    play_click_sfx()
                    set_cooldown()
                    print("[INFO] Play ditekan =  masuk menu kategori")
                    state = STATE_CATEGORY

        # ---------- STATE CATEGORY ----------
        elif state == STATE_CATEGORY:
            cat_header_x = (w_frame - kategori_header_btn.shape[1]) // 2
            frame = overlay_png(frame, kategori_header_btn, cat_header_x, header_y)

            buttons = [
                ("hewan", cat_hewan_btn),
                ("buah",  cat_buah_btn),
                ("kendaraan", cat_kendaraan_btn),
            ]

            total_width = sum(btn.shape[1] for _, btn in buttons) + 60 * (len(buttons) - 1)
            start_x = (w_frame - total_width) // 2
            y = int(h_frame * 0.72)

            x = start_x
            for key, btn_img in buttons:
                frame = draw_button_with_effect(frame, btn_img, x, y, fingertip)

                if fingertip and can_interact():
                    if point_on_png_button(fingertip[0], fingertip[1], x, y, btn_img):
                        selected_category = key
                        play_click_sfx()
                        set_cooldown()
                        print(f"[INFO] Kategori dipilih: {selected_category}")

                        score = 0
                        questions = build_questions_for_category(selected_category)
                        current_q_index = 0

                        (
                            current_q,
                            correct_answer,
                            answer_buttons,
                            question_sound,
                            question_start_time,
                            audio_initial_played,
                            repeat_used,
                            question_answered,
                            question_transition_deadline,
                        ) = setup_question(selected_category, current_q_index, questions, ALL_ANSWER_ASSETS)

                        state = STATE_GAME

                x += btn_img.shape[1] + 60

            if selected_category:
                put_text_with_outline(
                    frame,
                    f"Terpilih: {selected_category}",
                    (40, h_frame - 40),
                    cv2.FONT_HERSHEY_DUPLEX,
                    1.0,
                    (255, 255, 255),
```

```
                    2,
                    (0, 0, 0)
                )

        # ---------- STATE GAME ----------
        elif state == STATE_GAME:
            put_text_with_outline(
                frame,
                f"Skor: {score}",
                (30, 60),
                cv2.FONT_HERSHEY_DUPLEX,
                1.0,
                (255, 255, 255),
                2,
                (0, 0, 0)
            )

            total_q = len(questions)
            put_text_with_outline(
                frame,
                f"Soal: {current_q_index+1}/{total_q}",
                (30, 110),
                cv2.FONT_HERSHEY_DUPLEX,
                0.9,
                (255, 255, 255),
                2,
                (0, 0, 0)
            )

            t_now = now_ms()
            if not audio_initial_played and question_sound is not None:
                if t_now - question_start_time >= 2000:
                    question_sound.play()
                    audio_initial_played = True

            # Opsi jawaban
            if answer_buttons:
                gap_x = 30
                total_width = sum(btn["img"].shape[1] for btn in answer_buttons) + \
                              gap_x * (len(answer_buttons) - 1)

                start_x = int((w_frame - total_width) / 2)
                y_btn   = int(h_frame * 0.65)

                x = start_x
                for btn in answer_buttons:
                    img  = btn["img"]
                    word = btn["word"]

                    frame = draw_button_with_effect(frame, img, x, y_btn, fingertip)

                    if (not question_answered) and fingertip and can_interact():
                        if point_on_png_button(fingertip[0], fingertip[1], x, y_btn, img):
                            play_click_sfx()
                            set_cooldown()
                            print(f"[JAWAB] {word}")

                            if word == correct_answer:
                                score += 1
                                play_correct()
                                print("[HASIL] Benar!")
                            else:
```

```
                                play_wrong()
                                print("[HASIL] Salah!")

                            question_answered = True
                            question_transition_deadline = now_ms() + QUESTION_DELAY_MS

                    x += img.shape[1] + gap_x

            # Tombol ulangi audio (sebelum dijawab)
            if (not question_answered) and (not repeat_used) and question_sound is not None:
                rx = w_frame - repeat_audio_btn.shape[1] - 30
                ry = 30

                frame = draw_button_with_effect(frame, repeat_audio_btn, rx, ry, fingertip)

                if fingertip and can_interact():
                    if point_on_png_button(fingertip[0], fingertip[1], rx, ry, repeat_audio_btn):
                        if audio_initial_played:
                            play_click_sfx()
                            question_sound.play()
                            repeat_used = True
                            set_cooldown()
                            print("[INFO] Audio diulang sekali.")

            # Setelah delay, pindah soal / result
            if question_answered and question_transition_deadline is not None:
                if now_ms() >= question_transition_deadline:
                    question_transition_deadline = None
                    current_q_index += 1

                    if current_q_index < len(questions):
                        (
                            current_q,
                            correct_answer,
                            answer_buttons,
                            question_sound,
                            question_start_time,
                            audio_initial_played,
                            repeat_used,
                            question_answered,
                            question_transition_deadline,
                        ) = setup_question(selected_category, current_q_index, questions,
    ALL_ANSWER_ASSETS)
                    else:
                        print("[INFO] Semua soal selesai.")
                        state = STATE_RESULT

        # ---------- STATE RESULT ----------
        elif state == STATE_RESULT:
            if not result_sound_played:
                if score >= 8:      # nilai di atas 7
                    play_applause()
                    print("[RESULT] Applause!")
                else:
                    play_boo()
                    print("[RESULT] Boo!")
                result_sound_played = True

            put_text_with_outline(
                frame,
                "Kuis Selesai!",
                (int(w_frame * 0.3), int(h_frame * 0.4)),
```

```
                cv2.FONT_HERSHEY_DUPLEX,
                1.4,
                (255, 255, 255),
                3,
                (0, 0, 0)
            )

            put_text_with_outline(
                frame,
                f"Skor Akhir: {score} / {len(questions)}",
                (int(w_frame * 0.23), int(h_frame * 0.5)),
                cv2.FONT_HERSHEY_DUPLEX,
                1.2,
                (255, 255, 255),
                2,
                (0, 0, 0)
            )

            put_text_with_outline(
                frame,
                "Tekan 'q' untuk keluar",
                (int(w_frame * 0.3), int(h_frame * 0.6)),
                cv2.FONT_HERSHEY_DUPLEX,
                0.9,
                (255, 255, 255),
                2,
                (0, 0, 0)
            )

            btn_x = (w_frame - play_again_btn.shape[1]) // 2
            btn_y = int(h_frame * 0.72)

            frame = draw_button_with_effect(frame, play_again_btn, btn_x, btn_y, fingertip)

            if fingertip and can_interact():
                if point_on_png_button(fingertip[0], fingertip[1], btn_x, btn_y, play_again_btn):
                    play_click_sfx()
                    set_cooldown()
                    print("[INFO] Mainkan lagi ditekan = kembali ke menu kategori")

                    score = 0
                    selected_category = None
                    questions = []
                    current_q_index = 0
                    current_q = None
                    correct_answer = None
                    question_sound = None
                    audio_initial_played = False
                    repeat_used = False
                    result_sound_played = False
                    question_answered = False
                    question_transition_deadline = None

                    state = STATE_CATEGORY

        # ========================
        # UPDATE BGM & TAMPILKAN FRAME
        # ========================

        update_bgm_for_state(state)

        cv2.imshow(WINDOW_NAME, frame)
```

```
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

    if AUDIO_OK:
        pygame.mixer.quit()
```

Kode 1: Captionnya tulis di sini class

Dari kode diatas, file memiliki beberapa fungsi-fungsi yaitu:

4.1.1 Fungsi Audio

Fungsi audio terdiri dari beberapa fungsi yaitu:

- play_click_sfx() : Memutar suara klik tombol.
- play_correct() : Memutar suara jawaban benar.
- play_wrong() : Memutar suara jawaban salah.
- update_bgm_for_state(state) : Memperbarui musik latar sesuai state permainan.

4.1.2 Fungsi Waktu dan Cooldown

Fungsi waktu dan cooldown terdiri dari beberapa fungsi yaitu:

- now_ms(): Balikkan waktu dalam milidetik, dipakai untuk cooldown tombol delay audio soal dan delay sebelum pindah soal.
- can_interact(): Mengembalikan True kalau sudah lewat 2 detik dari klik tombol terakhir.
- set_cooldown(): Mengatur waktu klik tombol terakhir ke waktu sekarang.

4.1.3 Fungsi Gambar & Tombol

Fungsi gambar dan tombol terdiri dari beberapa fungsi yaitu:

- load_button_image(path, size): Baca PNG tombol ukuran tertentu.
- overlay_png(bg, png, x, y): Menempelkan PNG transparan di atas frame kamera (BGR) dengan alpha blending
- draw_button_with_effect(frame, btn_img, base_x, base_y, fingertip, press_offset=8): Menggambar tombol dengan efek “naik” saat disentuh jari.
- point_on_png_button(cx, cy, x, y, png): Mengecek apakah koordinat (cx,cy) berada di area tombol PNG yang ditempel di (x,y) dengan memperhitungkan alpha PNG.
- put_text_with_ outline: Menulis teks dengan outline agar terlihat di latar apapun.

4.1.4 Fungsi load tombol jawaban

Fungsi load tombol jawaban terdiri dari beberapa fungsi yaitu:

- load_all_answer_assets(): Memuat semua asset gambar tombol jawaban. Kemudian disimpan di dictionary.

4.1.5 Fungsi Logika soal

Fungsi logika soal terdiri dari beberapa fungsi yaitu:

- build_questions_for_category(cat_key): Membangun daftar soal untuk kategori tertentu.
- setup_question(cat_key, index): Menyiapkan soal berdasarkan kategori dan indeks soal.

4.1.6 Alur Program

- Program dijalankan dengan melakukan inisialisasi library, pendefinisian path aset gambar dan audio, penetapan state permainan, pengaturan cooldown input, serta pemuatan data soal untuk setiap kategori permainan.
- Sistem audio diinisialisasi menggunakan *pygame.mixer*, mencakup pemuatan backsound menu, sound effect klik tombol, jawaban benar, dan jawaban salah, serta penanganan kondisi ketika aset audio tidak tersedia.
- Seluruh aset gambar tombol dan header dimuat ke dalam memori menggunakan OpenCV dalam format BGRA agar mendukung transparansi dan deteksi sentuhan berbasis alpha channel.
- MediaPipe Hands dikonfigurasi untuk mendeteksi satu tangan dengan tingkat kepercayaan tertentu, sementara kamera diaktifkan menggunakan OpenCV dalam mode full-screen.
- Program memasuki *main loop* untuk membaca frame kamera secara real-time, memproses frame menjadi RGB, dan mendeteksi posisi jari telunjuk sebagai titik interaksi pengguna.
- Posisi jari telunjuk digunakan sebagai pengganti kursor untuk mendeteksi interaksi dengan tombol PNG melalui pengecekan area aktif dan nilai alpha pada pixel tombol.
- Sistem menampilkan antarmuka pengguna berdasarkan state permainan yang sedang aktif, seperti halaman awal, menu kategori, permainan, dan hasil akhir.
- Pada state *Home*, program menampilkan judul permainan dan tombol *Play*, memberikan efek visual saat tombol disentuh, serta berpindah ke menu kategori ketika tombol ditekan.
- Pada state *Category*, header *Pilih Kategori* ditampilkan bersama tombol pilihan kategori, kemudian sistem membangun daftar soal berdasarkan kategori yang dipilih dan menginisialisasi soal pertama.
- Pada state *Game*, program menampilkan skor dan nomor soal, memutar audio soal setelah jeda waktu tertentu, menampilkan opsi jawaban berbentuk tombol PNG, serta memberikan umpan balik suara untuk jawaban benar atau salah.
- Setelah sebuah soal dijawab, sistem menghentikan input sementara dan menerapkan delay transisi sebelum melanjutkan ke soal berikutnya atau masuk ke halaman hasil.
- Pada state *Result*, program menampilkan skor akhir, pesan penyelesaian kuis, serta tombol *Mainkan Lagi* untuk mengulang permainan dengan mengembalikan pemain ke menu kategori.
- Sistem cooldown berbasis waktu diterapkan secara global untuk mencegah input berulang meskipun jari pengguna tetap berada di area tombol.
- Backsound menu diputar secara otomatis pada state *Home* dan *Category*, serta dihentikan ketika memasuki state permainan dan hasil agar tidak mengganggu audio soal.
- Program terus berjalan hingga pengguna menekan tombol `q`, kemudian seluruh resource kamera, jendela OpenCV, dan sistem audio ditutup dengan aman.

4.2 File - Config

Secara konsep, file ini berfungsi sebagai tempat konfigurasi dan data statis serta sebagai pusat pengaturan supaya path aset tidak tersebar di banyak file, state game konsisten, data soal mudah dikelola, serta perubahan aset cukup dilakukan di satu tempat. Berikut ini adalah implementasi kode pada file **config.py**:

```python
import os

# ========================
# KONSTANTA & PATH ASSET
# ========================

HEADER_BUTTON_PATH = os.path.join("assets", "image", "buttons", "Guess The Word.png")
PLAY_BUTTON_PATH   = os.path.join("assets", "image", "buttons", "Play.png")
CAT_BUAH_PATH      = os.path.join("assets", "image", "buttons", "Kategori - Buah.png")
CAT_HEWAN_PATH     = os.path.join("assets", "image", "buttons", "Kategori - Hewan.png")
CAT_KENDARAAN_PATH = os.path.join("assets", "image", "buttons", "Kategori - Kendaraan.png")
PLAY_AGAIN_PATH    = os.path.join("assets", "image", "buttons", "Mainkan Lagi.png")
PILIH_KATEGORI_PATH = os.path.join("assets", "image", "buttons", "Pilih Kategori.png")

# Folder semua tombol jawaban
ANSWER_BUTTONS_DIR = os.path.join("assets", "image", "buttons")

# Button ulangi audio
REPEAT_AUDIO_PATH = os.path.join("assets", "image", "buttons", "Ulang Suara.png")

# Audio
BGM_MENU_PATH     = os.path.join("assets", "audio", "BG NCS.mp3")
SFX_CLICK_PATH    = os.path.join("assets", "audio", "Button Pressed.mp3")
SFX_CORRECT_PATH  = os.path.join("assets", "audio", "Correct.mp3")
SFX_WRONG_PATH    = os.path.join("assets", "audio", "Wrong.mp3")
SFX_APPLAUSE_PATH = os.path.join("assets", "audio", "Applause.mp3")
SFX_BOO_PATH      = os.path.join("assets", "audio", "Boo.mp3")

# Root audio soal
PITCH_AUDIO_ROOT = os.path.join("pitch_audio")

WINDOW_NAME = "Guess The Word Filter"

STATE_HOME     = "home"
STATE_CATEGORY = "category"
STATE_GAME     = "game"
STATE_RESULT   = "result"

# Cooldown berbasis waktu (ms)
BUTTON_COOLDOWN_MS = 2000   # 2 detik: setelah klik tombol, tombol lain tidak merespons
QUESTION_DELAY_MS  = 2500   # 2.5 detik: delay sebelum pindah ke soal berikutnya

# ========================
# DATA SOAL PER KATEGORI
# ========================

CATEGORY_DATA = {
    "buah": {
        "dir": os.path.join(PITCH_AUDIO_ROOT, "buah"),
        "id_words": ['apel', 'jeruk', 'pisang', 'mangga', 'semangka'],
        "en_words": ['strawberry', 'grape', 'papaya', 'kiwi', 'pineapple']
    },
    "kendaraan": {
        "dir": os.path.join(PITCH_AUDIO_ROOT, "kendaraan"),
        "id_words": ['mobil', 'motor', 'sepeda', 'bus', 'kereta'],
```

```
            "en_words": ['airplane', 'boat', 'helicopter', 'scooter', 'truck']
        },
        "hewan": {
            "dir": os.path.join(PITCH_AUDIO_ROOT, "hewan"),
            "id_words": ['kucing', 'anjing', 'gajah', 'harimau', 'singa'],
            "en_words": ['monkey', 'horse', 'dolphin', 'giraffe', 'rabbit']
        }
    }
```

Kode 2: Captionnya tulis di sini class

Berikut adalah penjelasan tentang alur yang ada pada file config.py:

4.2.1 Pengelolaan Path Aset Gambar

- Mendefinisikan lokasi seluruh aset gambar antarmuka seperti header judul permainan, tombol *Play*, tombol kategori, tombol *Mainkan Lagi*, dan tombol *Ulangi Audio*.
- Memastikan seluruh aset gambar tombol dapat diakses secara konsisten oleh modul lain tanpa melakukan *hardcode* path.
- Memudahkan proses perubahan atau penggantian aset visual tanpa perlu memodifikasi kode utama permainan.

4.2.2 Pengelolaan Path Aset Audio

- Menentukan lokasi backsound menu yang diputar pada halaman awal dan pemilihan kategori.
- Menyimpan path *sound effect* untuk interaksi pengguna seperti klik tombol, jawaban benar, jawaban salah, suara tepuk tangan (*applause*), dan suara cemoohan (*boo*).
- Memisahkan antara logika pemutaran audio dan konfigurasi lokasi file audio agar sistem lebih terstruktur.

4.2.3 Penentuan Direktori Audio Soal

- Mendefinisikan direktori utama audio soal hasil pemrosesan (*pitch-shifted audio*).
- Menjadi referensi dasar dalam membentuk path audio soal berdasarkan kategori dan kata yang dipilih secara acak.

4.2.4 Pendefinisian State Permainan

- Menetapkan state permainan seperti halaman awal (*home*), pemilihan kategori, tahap bermain, dan halaman hasil.
- Digunakan sebagai penentu alur tampilan dan logika interaksi pada *main loop*.
- Membantu pengendalian transisi antar tahap permainan agar tidak terjadi konflik alur.

4.2.5 Pengaturan Parameter Waktu dan Cooldown

- Menentukan durasi jeda interaksi tombol untuk mencegah input berulang akibat deteksi jari yang terus berada di atas tombol.
- Menentukan durasi jeda antar soal setelah pemain menjawab untuk meningkatkan kenyamanan dan kejelasan permainan.
- Memungkinkan penyesuaian responsivitas permainan tanpa mengubah logika utama.

4.2.6 Penyimpanan Data Soal per Kategori

- Menyimpan data kata untuk setiap kategori permainan.
- Memisahkan daftar soal Bahasa Indonesia dan Bahasa Inggris untuk menjamin proporsi soal yang seimbang.
- Menyediakan struktur data terpusat yang digunakan untuk pembuatan soal secara acak pada tahap permainan.

4.3 File - Audio Manager

File `audio_manager.py` berfungsi sebagai modul terpusat untuk mengatur seluruh fitur audio pada permainan, mulai dari inisialisasi *audio engine*, pemuatan berkas suara, hingga pemutaran *sound effect* dan *background music* sesuai dengan state permainan. Berikut ini adalah implementasi kode pada file **audio_manager.py**:

```

import os
import pygame

from .config import (
    BGM_MENU_PATH,
    SFX_CLICK_PATH,
    SFX_CORRECT_PATH,
    SFX_WRONG_PATH,
    STATE_HOME,
    STATE_CATEGORY,
    SFX_APPLAUSE_PATH,
    SFX_BOO_PATH
)

AUDIO_OK = False
bgm_available = False
click_sfx = None
correct_sfx = None
wrong_sfx = None
applause_sfx = None
boo_sfx = None


# ========================
# INIT AUDIO
# ========================

try:
    pygame.mixer.init()
    AUDIO_OK = True

    if os.path.exists(BGM_MENU_PATH):
        pygame.mixer.music.load(BGM_MENU_PATH)
        bgm_available = True
    else:
        print("[AUDIO] File BGM tidak ditemukan, skip backsong.")

    if os.path.exists(SFX_CLICK_PATH):
        click_sfx = pygame.mixer.Sound(SFX_CLICK_PATH)
    else:
        print("[AUDIO] File SFX click tidak ditemukan.")

    if os.path.exists(SFX_CORRECT_PATH):
```

```
            correct_sfx = pygame.mixer.Sound(SFX_CORRECT_PATH)
        else:
            print("[AUDIO] File SFX benar tidak ditemukan.")

        if os.path.exists(SFX_WRONG_PATH):
            wrong_sfx = pygame.mixer.Sound(SFX_WRONG_PATH)
        else:
            print("[AUDIO] File SFX salah tidak ditemukan.")

        if os.path.exists(SFX_APPLAUSE_PATH):
            applause_sfx = pygame.mixer.Sound(SFX_APPLAUSE_PATH)
        else:
            print("[AUDIO] File applause tidak ditemukan.")

        if os.path.exists(SFX_BOO_PATH):
            boo_sfx = pygame.mixer.Sound(SFX_BOO_PATH)
        else:
            print("[AUDIO] File boo tidak ditemukan.")

    except Exception as e:
        print("[AUDIO] Gagal inisialisasi audio:", e)
        AUDIO_OK = False


    def play_click_sfx():
        if AUDIO_OK and click_sfx is not None:
            click_sfx.play()


    def play_correct():
        if AUDIO_OK and correct_sfx is not None:
            correct_sfx.play()


    def play_wrong():
        if AUDIO_OK and wrong_sfx is not None:
            wrong_sfx.play()

    def play_applause():
        if AUDIO_OK and applause_sfx is not None:
            applause_sfx.play()

    def play_boo():
        if AUDIO_OK and boo_sfx is not None:
            boo_sfx.play()

    def update_bgm_for_state(state):
        if not (AUDIO_OK and bgm_available):
            return

        if state in (STATE_HOME, STATE_CATEGORY):
            if not pygame.mixer.music.get_busy():
                pygame.mixer.music.play(-1)  # loop
        else:
            if pygame.mixer.music.get_busy():
                pygame.mixer.music.stop()
```

Kode 3: Captionnya tulis di sini class

4.3.1 Pendefinisian Variabel Global Audio

- Mendefinisikan variabel `AUDIO_OK` sebagai penanda apakah inisialisasi `pygame.mixer` berhasil dilakukan atau tidak.
- Menggunakan variabel `bgm_available` untuk menyimpan status ketersediaan berkas *background music* (BGM) menu.
- Menyediakan variabel global `click_sfx`, `correct_sfx`, `wrong_sfx`, `applause_sfx`, dan `boo_sfx` untuk menyimpan objek *sound effect* yang sudah diload.
- Variabel-variabel ini digunakan oleh fungsi-fungsi pemutar audio sehingga dapat diakses secara global dari dalam modul.

4.3.2 Inisialisasi *Mixer* dan Pemuatan Berkas Audio

- Melakukan inisialisasi modul `pygame.mixer` di dalam blok `try-except` untuk mencegah program berhenti ketika terjadi error terkait audio.
- Jika inisialisasi berhasil, `AUDIO_OK` diset menjadi `True` sebagai indikator bahwa fitur audio siap digunakan.
- Mengecek keberadaan berkas BGM pada path `BGM_MENU_PATH`. Jika ditemukan, berkas tersebut diload ke `pygame.mixer.music` dan variabel `bgm_available` diset menjadi `True`.
- Memuat berbagai *sound effect* dari path yang telah didefinisikan di file konfigurasi, yaitu:
  - `SFX_CLICK_PATH` untuk suara klik tombol.
  - `SFX_CORRECT_PATH` untuk suara jawaban benar.
  - `SFX_WRONG_PATH` untuk suara jawaban salah.
  - `SFX_APPLAUSE_PATH` untuk suara tepuk tangan.
  - `SFX_BOO_PATH` untuk suara cemoohan.
- Jika salah satu berkas tidak ditemukan, modul akan menampilkan pesan informasi ke terminal, misalnya *"File SFX click tidak ditemukan"*, untuk mempermudah proses debugging.
- Jika terjadi exception selama proses inisialisasi, `AUDIO_OK` diset menjadi `False` dan fitur audio tidak akan digunakan oleh fungsi-fungsi lain.

4.3.3 Fungsi Pemutaran *Sound Effect*

- `play_click_sfx()`:
  - Memutar suara klik tombol ketika pemain menyentuh atau menekan sebuah tombol pada antarmuka permainan.
  - Hanya akan memutar suara jika `AUDIO_OK` bernilai `True` dan objek `click_sfx` tidak bernilai `None`.
- `play_correct()`:
  - Digunakan untuk memutar *sound effect* ketika pemain menjawab soal dengan benar.
  - Memberikan umpan balik positif secara audio agar permainan terasa lebih interaktif.
- `play_wrong()`:

  - Digunakan untuk memutar *sound effect* ketika jawaban pemain salah.
  - Membantu memberikan penegasan bahwa pilihan yang diambil tidak sesuai dengan jawaban yang benar.

- `play_applause()`:
  - Memutar suara tepuk tangan sebagai bentuk apresiasi, misalnya ketika skor akhir pemain tinggi atau melampaui ambang batas tertentu.

- `play_boo()`:
  - Memutar suara cemoohan (boo) ketika skor akhir pemain rendah, sehingga memberi nuansa humor atau feedback tambahan di layar hasil.

- Seluruh fungsi *sound effect* selalu melakukan pengecekan `AUDIO_OK` dan memastikan objek *sound* terkait tidak `None` sebelum memanggil `.play()`.

4.3.4 Fungsi Pengaturan *Background Music* Berdasarkan State

- Fungsi `update_bgm_for_state(state)` mengatur kapan BGM dinyalakan atau dimatikan berdasarkan state permainan saat ini.
- Jika audio tidak tersedia (`AUDIO_OK = False`) atau BGM tidak berhasil diload (`bgm_available = False`), fungsi ini akan langsung keluar tanpa melakukan apa pun.
- BGM hanya diputar ketika permainan berada pada state `STATE_HOME` atau `STATE_CATEGORY`, yaitu di menu utama dan menu pemilihan kategori.
- Pada state lain seperti `STATE_GAME` dan `STATE_RESULT`, BGM akan dihentikan jika sebelumnya sedang diputar, sehingga tidak mengganggu fokus pemain pada saat menjawab soal.
- Fungsi ini memastikan bahwa pengelolaan BGM tersentralisasi dan tidak perlu dipanggil secara manual di setiap perubahan state, cukup dengan memanggil `update_bgm_for_state()` di loop utama.

4.4 File - Audio Code

File `audio_code.py` berfungsi sebagai modul untuk menghasilkan file audio dari teks menggunakan layanan text-to-speech (TTS) Google melalui library `gTTS`. Modul ini menyediakan fungsi untuk membuat audio soal berdasarkan kata-kata yang telah ditentukan dalam data soal. Berikut ini adalah implementasi kode pada file **audio_code.py**:

```


import os
from gtts import gTTS
from pydub import AudioSegment

# Definisi struktur data, kata, dan path
KATEGORI = {
    'buah': {
        'path': 'data_audio/buah',
        'id_words': ['apel', 'jeruk', 'pisang', 'mangga', 'semangka'],
        'en_words': ['strawberry', 'grape', 'papaya', 'kiwi', 'pineapple']
    },
    'kendaraan': {
        'path': 'data_audio/kendaraan',
        'id_words': ['mobil', 'motor', 'sepeda', 'bus', 'kereta'],
```

```
            'en_words': ['airplane', 'boat', 'helicopter', 'scooter', 'truck']
        },
        'hewan': {
            'path': 'data_audio/hewan',
            'id_words': ['kucing', 'anjing', 'gajah', 'harimau', 'singa'],
            'en_words': ['monkey', 'horse', 'dolphin', 'giraffe', 'rabbit']
        }
    }

    def buat_audio_wav(teks, bahasa, file_path_wav):

        # Path sementara untuk file .mp3
        file_path_mp3 = file_path_wav.replace('.wav', '.mp3')

        try:
            # Buat audio .mp3 dengan gTTS
            tts = gTTS(text=teks, lang=bahasa, slow=False)
            tts.save(file_path_mp3)

            # Konversi .mp3 ke .wav menggunakan pydub
            audio = AudioSegment.from_mp3(file_path_mp3)
            audio.export(file_path_wav, format="wav")

            # Hapus file .mp3 sementara
            os.remove(file_path_mp3)

            print(f"  > Berhasil membuat: {file_path_wav}")

        except FileNotFoundError:
            print("\n[ERROR] FFmpeg tidak ditemukan!")
            print("Pastikan FFmpeg sudah terinstal dan ada di PATH sistem Anda.")
            # Hentikan eksekusi jika FFmpeg tidak ada
            exit()
        except Exception as e:
            print(f"  > Gagal membuat {file_path_wav}: {e}")
            # Hapus file mp3 jika ada error
            if os.path.exists(file_path_mp3):
                os.remove(file_path_mp3)


    print("Memulai proses pembuatan audio...")

    for nama_kategori, data in KATEGORI.items():

        output_dir = data['path']
        print(f"\nMemproses kategori: '{nama_kategori}' -> (di folder: {output_dir})")

        # Buat direktori jika belum ada
        os.makedirs(output_dir, exist_ok=True)

        # Buat audio untuk NAMA KATEGORI (kata representasi)
        # Ini untuk "satu kata yang merepresentasikan dari kategori tersebut"
        file_wav = os.path.join(output_dir, f"_{nama_kategori}.wav") # Diberi prefix _ agar mudah dibedakan
        buat_audio_wav(nama_kategori, 'id', file_wav)

        # Buat audio untuk kata-kata Bahasa Indonesia
        print("  --- Bahasa Indonesia ---")
        for kata in data['id_words']:
            file_wav = os.path.join(output_dir, f"{kata}.wav")
            buat_audio_wav(kata, 'id', file_wav)

```

```
        # 4. Buat audio untuk kata-kata Bahasa Inggris
        print("  --- Bahasa Inggris ---")
        for kata in data['en_words']:
            file_wav = os.path.join(output_dir, f"{kata}.wav")
            buat_audio_wav(kata, 'en', file_wav)

    print("Selesai! Semua file audio telah dibuat.")
```

Kode 4: Captionnya tulis di sini class

4.4.1 Struktur Data KATEGORI

- File `audio_code.py` diawali dengan pendefinisian struktur data `KATEGORI` dalam bentuk *dictionary* Python.
- Setiap entri `KATEGORI` merepresentasikan satu kategori kata, yaitu:
  - `'buah'`
  - `'kendaraan'`
  - `'hewan'`
- Untuk setiap kategori disimpan informasi:
  - `path`: lokasi folder output tempat file audio kategori tersebut akan disimpan (misalnya `data_audio/buah`).
  - `id_words`: daftar kata dalam Bahasa Indonesia yang merepresentasikan kategori tersebut, misalnya untuk kategori `buah` berisi `['apel', 'jeruk', 'pisang']`.
  - `en_words`: daftar kata dalam Bahasa Inggris yang masih terkait kategori yang sama, misalnya `['strawberry', 'grape', 'papaya']`.
- Struktur `KATEGORI` ini menjadi sumber data utama yang akan di-*loop* untuk membangkitkan file audio `.wav` untuk setiap kata yang dibutuhkan.

4.4.2 Fungsi `buat_audio_wav()`

- Fungsi `buat_audio_wav(teks, bahasa, file_path_wav)` bertugas membangkitkan satu file audio `.wav` dari teks tertentu menggunakan layanan `gTTS` dan `pydub`.
- Parameter yang digunakan:
  - `teks`: kata atau kalimat yang akan diubah menjadi suara.
  - `bahasa`: kode bahasa untuk `gTTS` (misalnya `'id'` untuk Bahasa Indonesia atau `'en'` untuk Bahasa Inggris).
  - `file_path_wav`: lokasi file `.wav` akhir yang akan disimpan.
- Di dalam fungsi, pertama-tama dibuat path sementara untuk file `.mp3` dengan cara:
  - Mengganti ekstensi `.wav` pada `file_path_wav` menjadi `.mp3` dan menyimpannya sebagai `file_path_mp3`.
- Proses utama fungsi:
  - Menggunakan `gTTS` untuk membuat file audio `.mp3` dari `teks` dan `bahasa` yang diberikan.
  - Menyimpan hasil `gTTS` ke `file_path_mp3`.

  - Membaca file `.mp3` menggunakan `AudioSegment.from_mp3()` dari pustaka `pydub`.
  - Mengekspor audio yang sudah dibaca ke format `.wav` menggunakan `audio.export(file_path_wav, format="wav")`.
  - Menghapus file `.mp3` sementara dengan `os.remove(file_path_mp3)` agar direktori output tetap bersih.
  - Menampilkan pesan *log* bahwa file `.wav` berhasil dibuat.

- Penanganan kesalahan pada fungsi:
  - Jika terjadi `FileNotFoundError` (umumnya karena FFmpeg tidak terpasang atau tidak ada di PATH), fungsi menampilkan pesan error yang menjelaskan bahwa FFmpeg harus diinstal, kemudian menghentikan program dengan `exit()`.
  - Jika terjadi `Exception` lain, fungsi menampilkan pesan kegagalan beserta detail error.
  - Jika error muncul setelah file `.mp3` dibuat, fungsi akan mencoba menghapus file `.mp3` sementara agar tidak tertinggal.

4.4.3 Alur Proses Utama Pembuatan Audio

- Program menampilkan pesan awal `"Memulai proses pembuatan audio..."` sebagai indikator bahwa proses generasi audio dimulai.
- Program melakukan *loop* terhadap setiap entri `KATEGORI` menggunakan:
  - `for nama_kategori, data in KATEGORI.items():`
- Untuk setiap kategori:
  - Mengambil nilai `data['path']` dan menyimpannya ke `output_dir` sebagai direktori tujuan penyimpanan file audio.
  - Menampilkan informasi kategori yang sedang diproses beserta nama folder outputnya.
  - Membuat direktori `output_dir` jika belum ada menggunakan `os.makedirs(output_dir, exist_ok=True)`.
- Program kemudian membangkitkan beberapa jenis file audio:
  - Audio untuk **nama kategori** itu sendiri, misalnya `"buah"`, `"kendaraan"`, dan `"hewan"`, yang disimpan dengan nama file `_{nama_kategori}.wav` (diberi prefix garis bawah agar mudah dibedakan dengan kata-kata biasa).
  - Audio untuk setiap kata Bahasa Indonesia yang terdapat pada `data['id_words']` dengan bahasa `'id'`.
  - Audio untuk setiap kata Bahasa Inggris yang terdapat pada `data['en_words']` dengan bahasa `'en'`.
- Setiap file audio dibuat dengan memanggil fungsi `buat_audio_wav()` dengan parameter:
  - `teks` diisi dengan kata yang sedang diproses.
  - `bahasa` diisi `'id'` atau `'en'` sesuai kategori kata.
  - `file_path_wav` diisi path file output di dalam folder kategori.
- Setelah seluruh kategori dan kata selesai diproses, program menampilkan pesan akhir:
  - Pesan sukses `"Selesai! Semua file audio telah dibuat."`.

4.5 File - Audio Processing

File `audio_processing.py` berfungsi sebagai modul untuk memproses file audio yang telah dihasilkan oleh `audio_code.py` dengan menerapkan teknik *pitch shifting* menggunakan pustaka `librosa` dan `soundfile`. Modul ini bertujuan untuk mengubah nada suara pada file audio agar menjadi lebih tinggi agar menambah tingkat kesulitan dalam menebak kata pada permainan. Berikut ini adalah implementasi kode pada file **audio_processing.py**:

```python

import os
import librosa
import soundfile as sf
import numpy as np

# Path
input_root = os.path.join(os.getcwd(), 'data_audio')
output_root = os.path.join(os.getcwd(), 'pitch_audio')

PITCH_STEPS = 7        # Nilai pitch shift
NOISE_FACTOR = 0.02    # Seberapa keras noise-nya

print(f"Memulai pemrosesan audio dari: {input_root}")
print("-" * 50)

# Berjalan menelusuri semua folder dan file di dalam input_root
for root, dirs, files in os.walk(input_root):
    for filename in files:
        if filename.lower().endswith('.wav'):

            # Tentukan Path Input Lengkap
            input_path = os.path.join(root, filename)

            # Tentukan Struktur Folder Output
            # (Mencari tahu file ini ada di folder kategori mana, misal: 'buah')
            relative_path = os.path.relpath(root, input_root)
            output_dir = os.path.join(output_root, relative_path)

            # Buat folder output jika belum ada
            os.makedirs(output_dir, exist_ok=True)

            # Tentukan Path Output Lengkap
            output_filename = filename.replace('.wav', '_pitch.wav') # Opsional: ganti nama file
            output_path = os.path.join(output_dir, output_filename)

            try:

                # Load Audio
                # sr=None agar sample rate mengikuti file asli
                y, sr = librosa.load(input_path, sr=None)

                # Pitch Shift (Naikkan Nada)
                y_shifted = librosa.effects.pitch_shift(y, sr=sr, n_steps=PITCH_STEPS)

                # Beri Noise (White Noise)
                # Generate noise acak sepanjang durasi audio
                noise = np.random.randn(len(y_shifted))
                # Gabungkan audio hasil pitch dengan noise
                y_final = y_shifted + (NOISE_FACTOR * noise)

                # Simpan
                sf.write(output_path, y_final, sr)
```

```python

                    print(f"[BERHASIL] {relative_path}/{filename} -> {relative_path}/{output_filename}")

                except Exception as e:
                    print(f"[GAGAL]    {relative_path}/{filename}: {e}")

    print("-" * 50)
    print("Proses selesai. Semua file tersimpan di folder 'pitch_audio'.")
```

Kode 5: Captionnya tulis di sini class

4.5.1 Konfigurasi Path dan Konstanta Pemrosesan

- File `audio_processing.py` diawali dengan impor pustaka:
  - `os` untuk pengelolaan path dan traversal folder.
  - `librosa` untuk pemrosesan audio (load dan pitch shifting).
  - `soundfile` (`sf`) untuk menyimpan kembali audio ke format `.wav`.
  - `numpy` (`np`) untuk membuat dan mengolah *white noise*.
- Variabel `input_root` didefinisikan sebagai folder sumber audio asli, yaitu `data_audio` yang diletakkan di dalam direktori kerja saat ini (`os.getcwd()`).
- Variabel `output_root` didefinisikan sebagai folder tujuan hasil pemrosesan audio, yaitu `pitch_audio` di dalam direktori kerja saat ini.
- Konstanta `PITCH_STEPS` digunakan untuk menentukan seberapa besar nada audio dinaikkan (dalam satuan *semitone*); pada kode ini diset ke nilai `7`, artinya pitch dinaikkan 7 semitone.
- Konstanta `NOISE_FACTOR` mengatur seberapa kuat *white noise* yang ditambahkan ke sinyal audio; pada kode ini nilainya `0.02`, yang berarti noise cukup halus dan hanya sebagai efek tambahan.
- Program menampilkan informasi awal mengenai folder sumber dengan mencetak:
  - `Memulai pemrosesan audio dari: ...`

4.5.2 Alur Proses Utama Pemrosesan Audio

- Program menelusuri seluruh folder dan file di dalam `input_root` menggunakan:
  - `for root, dirs, files in os.walk(input_root):`
- Untuk setiap berkas di dalam `files`, program melakukan pengecekan:
  - Hanya file dengan ekstensi `.wav` (diperiksa dengan `filename.lower().endswith('.wav')`) yang akan diproses.
- Jika file `.wav` ditemukan, program menentukan:
  - `input_path` sebagai path lengkap file input dengan `os.path.join(root, filename)`.
  - `relative_path` menggunakan `os.path.relpath(root, input_root)` untuk mengetahui struktur subfolder relatif dari `input_root` (misalnya `buah`, `hewan`, dan sebagainya).
  - `output_dir` yaitu folder tujuan di `output_root` yang mengikuti struktur folder yang sama dengan input, menggunakan `os.path.join(output_root, relative_path)`.

- Program memastikan folder tujuan sudah ada dengan:
  - `os.makedirs(output_dir, exist_ok=True)` sehingga jika folder belum ada akan dibuat, dan jika sudah ada tidak menyebabkan error.
- Nama file output ditentukan dengan menambahkan suffix `_pitch` sebelum ekstensi, menggunakan:
  - `output_filename = filename.replace('.wav', '_pitch.wav')`
  - `output_path = os.path.join(output_dir, output_filename)`
- Seluruh proses pemrosesan audio untuk satu file dibungkus dalam blok `try-except` untuk menangani kemungkinan error.
- Di dalam blok `try`, langkah-langkah pemrosesan audio adalah:
  - Memuat file audio menggunakan:
    - `y, sr = librosa.load(input_path, sr=None)`
    - `y` adalah array numpy berisi sampel audio.
    - `sr` adalah *sample rate*; `sr=None` berarti menggunakan sample rate asli file.
  - Menaikkan nada (*pitch shift*) audio dengan:
    - `y_shifted = librosa.effects.pitch_shift(y, sr=sr, n_steps=PITCH_STEPS)`
    - Fungsi ini menggeser frekuensi sinyal audio sebesar `PITCH_STEPS` semitone ke atas.
  - Membuat *white noise* dengan panjang yang sama dengan sinyal audio hasil pitch shift:
    - `noise = np.random.randn(len(y_shifted))`
  - Menggabungkan sinyal audio yang sudah dipitch dengan noise:
    - `y_final = y_shifted + (NOISE_FACTOR * noise)`
    - Faktor `NOISE_FACTOR` mengontrol seberapa besar kontribusi noise terhadap sinyal akhir.
  - Menyimpan hasil akhir ke file `.wav` baru menggunakan:
    - `sf.write(output_path, y_final, sr)`
    - `output_path` adalah lokasi file output.
    - `y_final` adalah sinyal audio akhir (pitch naik + noise).
    - `sr` adalah sample rate yang sama dengan audio input.
  - Menampilkan pesan keberhasilan:
    - `[BERHASIL] relative_path/filename -> relative_path/output_filename`
- Jika terjadi error apapun selama proses, blok `except Exception as e` akan:
  - Menampilkan pesan kegagalan dengan format:
  - `[GAGAL] relative_path/filename: e`
  - Sehingga pengguna mengetahui file mana yang bermasalah dan apa pesan error-nya.
- Setelah seluruh file di semua folder diproses, program menampilkan garis pembatas dan pesan penutup:
  - `"Proses selesai. Semua file tersimpan di folder 'pitch_audio'."`

4.5.3 Fungsi dan Pustaka Utama yang Digunakan

- **Fungsi-fungsi dari modul `os`:**
  - `os.getcwd()` digunakan untuk mendapatkan direktori kerja saat ini sebagai basis pembuatan path input dan output.
  - `os.path.join()` digunakan untuk menggabungkan nama folder dan file menjadi path lengkap yang sesuai sistem operasi.
  - `os.walk()` digunakan untuk menelusuri seluruh subdirektori dan file di dalam `input_root`.
  - `os.path.relpath()` digunakan untuk mendapatkan path relatif terhadap `input_root`, sehingga struktur folder output bisa mengikuti struktur folder input.
  - `os.makedirs(..., exist_ok=True)` digunakan untuk membuat folder output jika belum ada.
- **Fungsi dari `librosa`:**
  - `librosa.load()` membaca file audio menjadi array numpy dan mengembalikan `(y, sr)`.
  - `librosa.effects.pitch_shift()` mengubah tinggi nada (pitch) sinyal audio tanpa mengubah durasi secara signifikan.
- **Fungsi dari `numpy`:**
  - `np.random.randn(len(y_shifted))` menghasilkan deret *random* dengan distribusi normal (white noise) dengan panjang yang sama dengan sinyal audio.
  - Operasi penjumlahan dan perkalian skalar pada array digunakan untuk mencampur noise dengan sinyal audio.
- **Fungsi dari `soundfile`:**
  - `sf.write(output_path, y_final, sr)` menyimpan array numpy audio ke dalam file `.wav` di path yang ditentukan dengan sample rate tertentu.

4.6 File - Graphic Manager

File `graphic.py` berfungsi sebagai modul untuk mengelola pemuatan dan penampilan aset gambar pada antarmuka permainan menggunakan pustaka `OpenCV`. Modul ini menyediakan fungsi untuk memuat gambar dari file, menampilkan gambar pada jendela OpenCV, serta mengatur posisi dan ukuran gambar sesuai kebutuhan. Berikut ini adalah implementasi kode pada file **graphic.py**:

```python

import os
import cv2
import numpy as np

from .config import ANSWER_BUTTONS_DIR

# ========================
# FUNGSI GRAFIS & BUTTON
# ========================

def load_button_image(path, size):
    w, h = size
    img = cv2.imread(path, cv2.IMREAD_UNCHANGED)

    if img is None:
        dummy = np.zeros((h, w, 4), dtype=np.uint8)
```

```python
            dummy[..., :3] = (0, 0, 255)
            dummy[..., 3] = 255
            cv2.putText(
                dummy,
                "BTN",
                (10, int(h * 0.65)),
                cv2.FONT_HERSHEY_DUPLEX,
                1,
                (255, 255, 255, 255),
                2,
                cv2.LINE_AA,
            )
            return dummy

        img = cv2.resize(img, (w, h))
        if img.shape[2] == 3:
            bgr = img
            alpha = np.full((h, w, 1), 255, dtype=np.uint8)
            img = np.concatenate([bgr, alpha], axis=2)
        return img


    def overlay_png(bg, png, x, y):
        h, w, _ = png.shape

        if x < 0 or y < 0 or x + w > bg.shape[1] or y + h > bg.shape[0]:
            return bg

        roi = bg[y:y + h, x:x + w]

        bgr = png[..., :3]
        alpha = png[..., 3:] / 255.0

        roi = (alpha * bgr + (1 - alpha) * roi).astype(np.uint8)
        bg[y:y + h, x:x + w] = roi
        return bg


    def point_on_png_button(cx, cy, x, y, png):
        h, w, _ = png.shape

        if not (x <= cx < x + w and y <= cy < y + h):
            return False

        local_x = cx - x
        local_y = cy - y
        alpha = png[local_y, local_x, 3]
        return alpha > 10


    def draw_button_with_effect(frame, btn_img, base_x, base_y, fingertip,
                                press_offset=8):
        """
        Gambar tombol. Kalau jari menyentuh tombol, tombol digambar sedikit
        lebih atas (press_offset pixel).
        Hitbox tetap di posisi base_x, base_y.
        """
        y_draw = base_y
        if fingertip is not None:
            if point_on_png_button(fingertip[0], fingertip[1], base_x, base_y, btn_img):
                y_draw = base_y - press_offset  # efek naik

```

```python
        frame = overlay_png(frame, btn_img, base_x, y_draw)
        return frame


    def put_text_with_outline(
        img,
        text,
        org,
        font,
        font_scale,
        color,
        thickness,
        outline_color=(0, 0, 0),
        outline_thickness=None
    ):
        x, y = org
        if outline_thickness is None:
            outline_thickness = thickness + 2

        for dx, dy in [(-2, -2), (-2, 2), (2, -2), (2, 2)]:
            cv2.putText(
                img,
                text,
                (x + dx, y + dy),
                font,
                font_scale,
                outline_color,
                outline_thickness,
                cv2.LINE_AA,
            )
        cv2.putText(
            img,
            text,
            (x, y),
            font,
            font_scale,
            color,
            thickness,
            cv2.LINE_AA,
        )
        return img


    def load_all_answer_assets():
        all_assets = {
            "hewan": [],
            "buah": [],
            "kendaraan": []
        }

        if not os.path.isdir(ANSWER_BUTTONS_DIR):
            print("[ASSET] Folder tombol jawaban tidak ditemukan:", ANSWER_BUTTONS_DIR)
            return all_assets

        for fname in os.listdir(ANSWER_BUTTONS_DIR):
            if not fname.lower().endswith(".png"):
                continue

            name_no_ext = os.path.splitext(fname)[0]
            parts = name_no_ext.split("-")
            if len(parts) < 2:
                continue
```

```python

            cat_label = parts[0].strip().lower()
            word = parts[1].strip().lower()

            if cat_label in all_assets:
                path = os.path.join(ANSWER_BUTTONS_DIR, fname)
                img = load_button_image(path, (290, 95))
                all_assets[cat_label].append({
                    "word": word,
                    "img": img
                })

        for k, v in all_assets.items():
            print(f"[ASSET] Kategori {k} punya {len(v)} tombol jawaban.")
        return all_assets
```

Kode 6: Captionnya tulis di sini class

4.6.1 Gambaran Umum File `graphics.py`

- Berkas `graphics.py` berisi fungsi-fungsi utilitas grafis yang digunakan untuk:
  - Memuat gambar tombol dari aset (*button assets*) dengan dukungan kanal alpha (transparansi).
  - Menempelkan (overlay) gambar PNG transparan di atas frame kamera.
  - Mengecek apakah ujung jari (*fingertip*) menyentuh area tombol yang tidak transparan.
  - Memberikan efek animasi sederhana saat tombol disentuh (naik sedikit ke atas).
  - Menampilkan teks dengan outline (*stroke*) agar tetap terbaca di background yang ramai.
  - Memuat seluruh aset tombol jawaban dari folder dan mengelompokkan berdasarkan kategori (hewan, buah, kendaraan).
- Berkas ini menjadi lapisan abstraksi untuk semua operasi yang berhubungan dengan tampilan tombol dan teks, sehingga kode utama di `main.py` menjadi lebih bersih dan terstruktur.

4.6.2 Fungsi `load_button_image`

- **Tujuan:** Memuat gambar tombol dari file PNG dan memastikan gambar memiliki kanal alpha (BGRA) dengan ukuran tertentu.
- **Alur kerja:**
  - Menerima parameter `path` (lokasi file gambar) dan `size` (lebar, tinggi).
  - Membaca gambar dengan `cv2.imread(..., cv2.IMREAD_UNCHANGED)` agar kanal alpha (jika ada) ikut terbaca.
  - Jika gambar gagal dimuat (`img is None`):
    * Membuat gambar dummy berwarna merah dengan ukuran yang diminta.
    * Menambahkan tulisan `"BTN"` di atas gambar dummy sebagai indikator tombol placeholder.
  - Jika gambar berhasil dimuat:
    * Gambar di-*resize* ke ukuran `(w, h)`.
    * Jika gambar hanya memiliki 3 kanal (BGR), fungsi menambahkan kanal alpha penuh (255) sehingga menjadi BGRA.
  - Mengembalikan gambar tombol dalam format array `numpy` dengan 4 kanal (BGRA).

4.6.3 Fungsi `overlay_png`

- **Tujuan:** Menempelkan (overlay) gambar PNG dengan kanal alpha di atas frame background (gambar kamera) menggunakan alpha blending.
- **Alur kerja:**
  - Menerima parameter:
    - `bg`: frame background (BGR).
    - `png`: gambar tombol atau aset lain (BGRA).
    - `x, y`: posisi kiri-atas tempat gambar akan ditempel di frame.
  - Mengecek apakah posisi dan ukuran tombol masih berada di dalam batas frame; jika tidak, fungsi mengembalikan frame tanpa perubahan.
  - Mengambil *region of interest* (ROI) pada background sesuai ukuran gambar PNG.
  - Memisahkan kanal BGR dan kanal alpha dari gambar PNG.
  - Menghitung hasil campuran dengan rumus:
    - `roi = alpha * bgr + (1 - alpha) * roi`
  - Menulis kembali hasil ROI ke dalam background dan mengembalikan frame yang sudah ditempeli gambar PNG.

4.6.4 Fungsi `point_on_png_button`

- **Tujuan:** Mengecek apakah titik tertentu (misalnya posisi ujung jari) berada di atas area tombol yang tidak transparan.
- **Alur kerja:**
  - Menerima parameter:
    - `cx, cy`: koordinat titik (misalnya fingertip).
    - `x, y`: posisi kiri-atas tombol.
    - `png`: gambar tombol (BGRA).
  - Mengecek apakah titik berada di dalam bounding box tombol (dilihat dari `x`, `y`, lebar dan tinggi PNG).
  - Jika di luar bounding box, fungsi mengembalikan `False`.
  - Jika di dalam bounding box:
    - Menghitung koordinat lokal (`local_x`, `local_y`) relatif terhadap gambar tombol.
    - Mengambil nilai alpha pada piksel tersebut.
    - Mengembalikan `True` jika alpha lebih besar dari ambang (misalnya `10`), artinya piksel tersebut bukan area transparan; jika tidak, `False`.

4.6.5 Fungsi `draw_button_with_effect`

- **Tujuan:** Menggambar tombol dengan efek animasi sederhana: ketika jari menyentuh tombol, tampilan tombol sedikit naik ke atas (*hover/press effect*).
- **Alur kerja:**
  - Menerima parameter:
    - `frame`: frame kamera saat ini.

    * `btn_img`: gambar tombol (BGRA).
    * `base_x, base_y`: posisi dasar tombol di layar.
    * `fingertip`: koordinat ujung jari (bisa `None` jika tidak ada tangan terdeteksi).
    * `press_offset`: besar perpindahan tombol ke atas ketika disentuh (default 8 piksel).
  - Secara default, tombol digambar di posisi `(base_x, base_y)`.
  - Jika `fingertip` tidak `None` dan fungsi `point_on_png_button` mendeteksi bahwa titik tersebut menyentuh tombol:
    * Posisi gambar digeser ke atas sebesar `press_offset` (y lebih kecil).
  - Memanggil `overlay_png` untuk menggambar tombol di posisi baru (dengan efek naik).
  - Mengembalikan frame yang sudah berisi tombol dengan efek.

4.6.6 Fungsi `put_text_with_outline`

- **Tujuan:** Menampilkan teks di atas frame dengan outline (*stroke*) sehingga tulisan tetap terbaca meskipun background ramai atau terang.
- **Alur kerja:**
  - Menerima parameter:
    * `img`: frame atau gambar tempat teks akan digambar.
    * `text`: string teks yang ingin ditampilkan.
    * `org`: posisi baseline teks (`(x, y)`).
    * `font`: jenis font OpenCV (misalnya `cv2.FONT_HERSHEY_DUPLEX`).
    * `font_scale`: skala besar teks.
    * `color`: warna teks utama.
    * `thickness`: ketebalan teks utama.
    * `outline_color`: warna outline (default hitam).
    * `outline_thickness`: ketebalan outline (jika `None`, otomatis `thickness + 2`).
  - Menggambar outline dengan memanggil `cv2.putText` beberapa kali di sekitar posisi teks:
    * Menggeser sedikit posisi teks ke kombinasi `(-2,-2)`, `(-2,2)`, `(2,-2)`, dan `(2,2)`.
    * Menggunakan warna `outline_color` dan ketebalan `outline_thickness`.
  - Setelah outline digambar, fungsi menggambar teks utama di posisi `(x, y)` dengan warna dan ketebalan yang diinginkan.
  - Mengembalikan gambar `img` yang sudah berisi teks dengan outline.

4.6.7 Fungsi `load_all_answer_assets`

- **Tujuan:** Memuat seluruh aset tombol jawaban dari folder `ANSWER_BUTTONS_DIR` dan mengelompokkan berdasarkan kategori (`hewan`, `buah`, `kendaraan`).
- **Alur kerja:**
  - Inisialisasi struktur data:
    * `all_assets = {"hewan": [], "buah": [], "kendaraan": []}` untuk menampung daftar tombol per kategori.
  - Mengecek apakah folder `ANSWER_BUTTONS_DIR` ada:
    * Jika tidak ada, menampilkan pesan bahwa folder tidak ditemukan dan mengembalikan `all_assets` kosong.

- Melakukan iterasi untuk setiap file di dalam `ANSWER_BUTTONS_DIR`:
  - Hanya memproses file dengan ekstensi `.png`.
  - Mengambil nama file tanpa ekstensi, misalnya `"Hewan - Monkey"`.
  - Memecah nama berdasarkan tanda `'-'` menjadi dua bagian:
    - Bagian pertama sebagai label kategori (`"hewan"`, `"buah"`, `"kendaraan"`) setelah di-*strip* dan di-`lower()`.
    - Bagian kedua sebagai kata (`word`) jawaban, juga di-`strip` dan di-`lower()`.
  - Jika `cat_label` termasuk dalam kunci `all_assets`:
    - Menyusun path lengkap gambar tombol.
    - Memanggil `load_button_image` dengan ukuran `(290, 95)` untuk memuat gambar tombol.
    - Menambahkan dictionary `{"word": word, "img": img}` ke dalam list kategori yang sesuai.
- Setelah semua file diproses, fungsi mencetak jumlah tombol per kategori, misalnya:
  - `[ASSET] Kategori hewan punya X tombol jawaban.`
- Mengembalikan `all_assets`, yang kemudian digunakan oleh logika game untuk menampilkan tombol jawaban sesuai kategori dan kata yang benar.

4.7 File - Game Logic

File `game_logic.py` berfungsi sebagai modul yang mengelola logika permainan utama, termasuk pemilihan kategori, pemilihan kata acak, penilaian jawaban, dan pengaturan skor. Modul ini berinteraksi dengan modul grafis (`graphics.py`) untuk menampilkan tombol jawaban dan memberikan umpan balik kepada pemain. Berikut ini adalah implementasi kode pada file **game_logic.py**:

```python
import os
import random
import pygame

from .config import CATEGORY_DATA
from .audio_manager import AUDIO_OK
from .utils import now_ms

def build_questions_for_category(cat_key):
    """
    Bangun list 10 soal (5 ID, 5 EN) untuk kategori tertentu.
    """
    data = CATEGORY_DATA[cat_key]
    id_words = data["id_words"]
    en_words = data["en_words"]

    id_pick = random.sample(id_words, min(5, len(id_words)))
    en_pick = random.sample(en_words, min(5, len(en_words)))

    q_list = []
    for w in id_pick:
        q_list.append({
            "word": w.lower(),
            "lang": "id",
            "audio_path": os.path.join(data["dir"], f"{w}.wav")
        })
    for w in en_pick:
        q_list.append({
            "word": w.lower(),
            "lang": "en",
```

```
                "audio_path": os.path.join(data["dir"], f"{w}.wav")
            })

        random.shuffle(q_list)
        return q_list[:10]


    def setup_question(cat_key, index, questions, all_answer_assets):
        """
        Siapkan soal ke-index, load audio & opsi jawaban dari asset PNG.
        Mengembalikan:
          current_q, correct_answer, answer_buttons,
          question_sound, question_start_time,
          audio_initial_played, repeat_used,
          question_answered, question_transition_deadline
        """
        assets_for_cat = all_answer_assets.get(cat_key, [])
        if not assets_for_cat:
            print("[ASSET] Tidak ada asset jawaban untuk kategori:", cat_key)

        current_q = questions[index]
        correct_answer = current_q["word"]

        correct_btn = None
        wrong_btns = []
        for item in assets_for_cat:
            if item["word"] == correct_answer:
                correct_btn = item
            else:
                wrong_btns.append(item)

        if correct_btn is None:
            print(f"[WARN] Tidak ketemu asset untuk kata '{correct_answer}' di kategori '{cat_key}'
")
            answer_buttons = random.sample(assets_for_cat, min(4, len(assets_for_cat)))
        else:
            if len(wrong_btns) >= 3:
                wrong_btns = random.sample(wrong_btns, 3)
            answer_buttons = wrong_btns + [correct_btn]
            random.shuffle(answer_buttons)

        question_sound = None
        if AUDIO_OK and os.path.exists(current_q["audio_path"]):
            try:
                question_sound = pygame.mixer.Sound(current_q["audio_path"])
            except Exception as e:
                print("[AUDIO] Gagal load audio soal:", current_q["audio_path"], e)
        else:
            print("[AUDIO] File audio soal tidak ditemukan:", current_q["audio_path"])

        question_start_time = now_ms()
        audio_initial_played = False
        repeat_used = False

        question_answered = False
        question_transition_deadline = None

        print(f"[SOAL] Q{index+1}: {correct_answer} ({current_q['lang']})")

        return (
            current_q,
            correct_answer,
```

```
            answer_buttons,
            question_sound,
            question_start_time,
            audio_initial_played,
            repeat_used,
            question_answered,
            question_transition_deadline,
        )
```

Kode 7: Captionnya tulis di sini class

4.7.1 Gambaran Umum File `game_logic.py`

- Berkas `game_logic.py` berisi fungsi-fungsi yang menangani logika utama penyusunan soal dalam permainan.
- Logika yang diatur di file ini meliputi:
  - Membuat daftar soal untuk setiap kategori (*buah*, *kendaraan*, *hewan*).
  - Menentukan kata yang akan dijadikan jawaban benar dan opsi jawaban yang lain sebagai pengecoh.
  - Menyiapkan audio soal untuk setiap pertanyaan.
  - Mengembalikan seluruh state awal yang dibutuhkan sebelum soal ditampilkan di tampilan utama.
- Berkas ini memisahkan logika permainan (soal dan jawaban) dari tampilan UI sehingga kode lebih rapi dan mudah di-maintain.

4.7.2 Fungsi `build_questions_for_category`

- **Tujuan:** Menyusun daftar maksimal 10 soal untuk satu kategori tertentu (kombinasi Bahasa Indonesia dan Bahasa Inggris).
- **Parameter:**
  - `cat_key`: kunci kategori, misalnya `"buah"`, `"kendaraan"`, atau `"hewan"`.
- **Data sumber:**
  - Mengambil konfigurasi kategori dari `CATEGORY_DATA` yang diimpor dari `config.py`.
  - `CATEGORY_DATA[cat_key]` berisi:
    - `"dir"`: folder tempat file audio soal disimpan.
    - `"id_words"`: daftar kata Bahasa Indonesia.
    - `"en_words"`: daftar kata Bahasa Inggris.
- **Alur kerja:**
  - Mengambil daftar kata Bahasa Indonesia (`id_words`) dan Bahasa Inggris (`en_words`) dari kategori tersebut.
  - Mengambil sampel acak untuk masing-masing:
    - `id_pick`: maksimal 5 kata dari `id_words`.
    - `en_pick`: maksimal 5 kata dari `en_words`.
  - Menyusun list `q_list` yang berisi dictionary untuk setiap soal:

    * `"word"`: kata jawaban dalam huruf kecil.
    * `"lang"`: kode bahasa, `"id"` atau `"en"`.
    * `"audio_path"`: path lengkap ke file audio `.wav` untuk kata tersebut.
  - Menggabungkan soal Bahasa Indonesia dan Bahasa Inggris lalu mengacak urutannya dengan `random.shuffle`.
  - Mengembalikan maksimal 10 soal pertama dengan `return q_list[:10]`.

- **Hasil:**
  - Menghasilkan list soal yang siap digunakan oleh `main.py` untuk ditampilkan dan dimainkan.

4.7.3 Fungsi `setup_question`

- **Tujuan:** Menyiapkan seluruh data yang dibutuhkan untuk menampilkan satu soal tertentu:
  - Menentukan soal aktif (`current_q`).
  - Menentukan jawaban benar dan opsi jawaban lain dalam bentuk tombol gambar.
  - Menyiapkan audio soal yang akan diputar.
  - Menginisialisasi flag dan waktu yang berhubungan dengan logika soal (status terjawab, delay, dan lain-lain).
- **Parameter:**
  - `cat_key`: kategori soal yang sedang dimainkan (`"buah"`, `"kendaraan"`, atau `"hewan"`).
  - `index`: indeks soal ke berapa yang ingin disiapkan, misalnya 0 untuk soal pertama.
  - `questions`: list soal yang sudah dibangun sebelumnya oleh `build_questions_for_category`.
  - `all_answer_assets`: kumpulan aset tombol jawaban per kategori yang di-load oleh modul grafis (`graphics.py`).
- **Proses pemilihan jawaban:**
  - Mengambil kumpulan aset jawaban untuk kategori tersebut dari `all_answer_assets`.
  - Mengambil soal yang sesuai indeks dan menyimpan:
    * `current_q`: dictionary berisi kata, bahasa, dan `audio_path`.
    * `correct_answer`: kata jawaban benar dalam huruf kecil.
  - Melakukan loop pada semua aset gambar kategori untuk:
    * Mencari tombol gambar yang `word`-nya sama dengan `correct_answer` sebagai `correct_btn`.
    * Mengumpulkan tombol lain sebagai `wrong_btns`.
  - Jika aset jawaban benar tidak ditemukan:
    * Menampilkan peringatan dan memilih maksimal 4 tombol acak dari kategori tersebut sebagai opsi jawaban.
  - Jika aset jawaban benar ditemukan:
    * Memilih 3 tombol salah secara acak dari `wrong_btns` (jika jumlahnya cukup).
    * Menggabungkan 3 jawaban salah + 1 jawaban benar menjadi list `answer_buttons`.
    * Mengacak urutan `answer_buttons` agar posisi jawaban benar tidak selalu sama.
- **Proses persiapan audio:**
  - Menginisialisasi `question_sound` dengan `None`.

  - Jika `AUDIO_OK` bernilai `True` dan file audio soal ada di disk:
    * Mencoba memuat audio soal menggunakan `pygame.mixer.Sound`.
    * Jika gagal, menampilkan pesan error.
  - Jika file audio tidak ditemukan, menampilkan pesan bahwa audio soal tidak tersedia.

- **Inisialisasi status dan waktu soal:**
  - Mengatur `question_start_time` dengan memanggil `now_ms()` dari modul `utils` untuk mencatat kapan soal mulai.
  - Mengatur:
    * `audio_initial_played = False` agar sistem tahu audio awal belum diputar.
    * `repeat_used = False` agar tombol ulangi audio hanya bisa dipakai sekali.
    * `question_answered = False` sebagai penanda bahwa soal belum dijawab.
    * `question_transition_deadline = None` yang nanti akan diisi ketika soal sudah dijawab dan menunggu delay sebelum pindah soal.
  - Mencetak informasi soal di console, misalnya `[SOAL] Q1:  apel (id)`.

- **Nilai kembali:**
  - Fungsi ini mengembalikan beberapa nilai yang akan dipakai oleh `main.py` untuk mengatur state permainan:
    * `current_q`: data soal aktif.
    * `correct_answer`: kata jawaban benar.
    * `answer_buttons`: list tombol jawaban (berisi `word` dan `img`).
    * `question_sound`: objek `pygame.mixer.Sound` untuk audio soal (bisa `None`).
    * `question_start_time`: waktu mulai soal dalam milidetik.
    * `audio_initial_played`: flag audio awal.
    * `repeat_used`: flag apakah tombol ulangi audio sudah dipakai.
    * `question_answered`: status apakah soal sudah dijawab.
    * `question_transition_deadline`: waktu (timestamp) kapan boleh pindah ke soal berikutnya setelah delay.

4.8 File - Utils

File `utils.py` berfungsi sebagai modul utilitas yang menyediakan fungsi-fungsi pembantu umum yang digunakan di berbagai bagian program. Modul ini berisi fungsi untuk mendapatkan waktu saat ini dalam milidetik, yang berguna untuk mengatur timing dan delay dalam logika permainan. Berikut ini adalah implementasi kode pada file **utils.py**:

```python
import time
from .config import BUTTON_COOLDOWN_MS

last_hit_time = 0  # timestamp terakhir kena klik


def now_ms():
    return int(time.time() * 1000)


def can_interact():
    """True kalau sudah lewat cooldown tombol."""
    return now_ms() - last_hit_time >= BUTTON_COOLDOWN_MS
```

```


def set_cooldown():
    global last_hit_time
    last_hit_time = now_ms()
```

Kode 8: Captionnya tulis di sini class

4.8.1 Gambaran Umum File `utils.py`

- Berkas `utils.py` berfungsi sebagai modul utilitas umum yang menangani:
  - Manajemen waktu berbasis milidetik.
  - Pengendalian *cooldown* interaksi tombol.
- Modul ini digunakan untuk mencegah input ganda ketika jari pengguna berada terlalu lama di atas tombol.
- Dengan memisahkan logika waktu ke file ini, kode utama (`main.py`) menjadi lebih bersih dan terstruktur.

4.8.2 Variabel Global

- `last_hit_time`
  - Menyimpan waktu (timestamp dalam milidetik) terakhir kali terjadi interaksi tombol.
  - Digunakan sebagai acuan untuk menghitung apakah *cooldown* sudah berakhir atau belum.

4.8.3 Fungsi `now_ms`

- **Tujuan:** Mengambil waktu saat ini dalam satuan milidetik.
- **Proses:**
  - Menggunakan `time.time()` untuk mendapatkan waktu sekarang dalam detik.
  - Mengonversinya ke milidetik dengan mengalikan 1000.
- **Kegunaan dalam sistem:**
  - Digunakan untuk perhitungan *cooldown* tombol.
  - Dipakai juga pada logika delay soal dan audio.

4.8.4 Fungsi `can_interact`

- **Tujuan:** Mengecek apakah sistem sudah boleh menerima input tombol baru.
- **Logika kerja:**
  - Menghitung selisih waktu antara `now_ms()` dan `last_hit_time`.
  - Membandingkan selisih tersebut dengan konstanta `BUTTON_COOLDOWN_MS`.
- **Nilai kembali:**
  - `True` jika waktu cooldown sudah terlewati.
  - `False` jika cooldown masih aktif dan input harus diabaikan.

- **Peran dalam filter:**
  - Mencegah tombol tertekan berkali-kali saat jari masih menyentuh tombol.
  - Membuat interaksi terasa lebih stabil dan natural.

4.8.5 Fungsi `set_cooldown`

- **Tujuan:** Mengaktifkan ulang cooldown setelah tombol ditekan.
- **Proses:**
  - Mengisi ulang variabel global `last_hit_time` dengan waktu saat ini menggunakan `now_ms()`.
- **Penggunaan:**
  - Dipanggil setiap kali tombol berhasil ditekan (Play, Kategori, Jawaban, Mainkan Lagi, dll).
  - Menandai awal jeda waktu sebelum input berikutnya diizinkan.

4.8.6 Peran `utils.py` dalam Keseluruhan Sistem

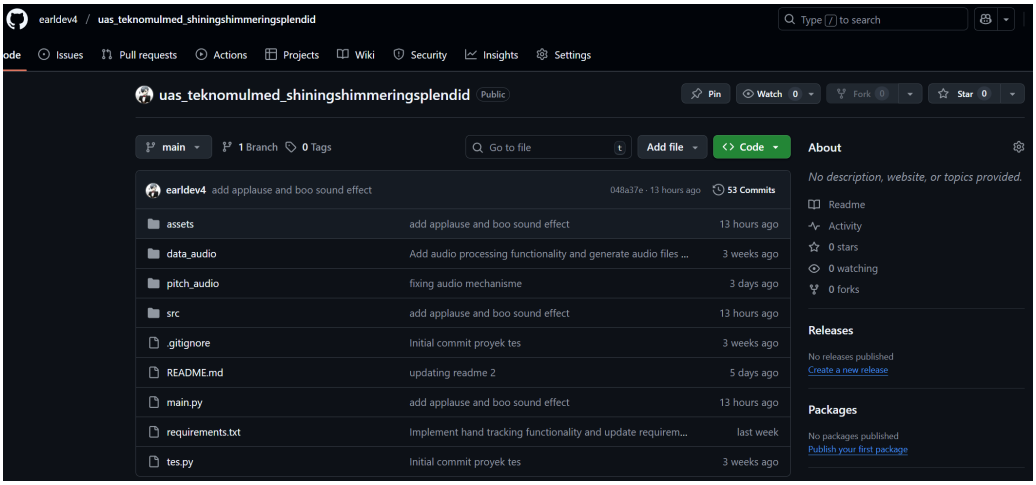
- Menjadi pusat kontrol logika waktu dan interaksi.
- Menjaga konsistensi perilaku tombol di seluruh state permainan:
  - Menu Home.
  - Menu Pilih Kategori.
  - Tahap Gameplay.
  - Menu Result.
- Membantu meningkatkan pengalaman pengguna dengan mencegah input tidak disengaja.

5 Implementasi Program

5.1 Menyalin Repository dari GitHub

Langkah pertama dalam mengimplementasikan program ini adalah menyalin URL repository dari GitHub. Berikut adalah langkah-langkahnya:

1. Buka halaman repository di GitHub melalui tautan berikut: Link Repository.
2. Klik tombol "Code" yang berwarna hijau di sebelah kanan atas halaman repository.
3. Salin URL repository yang muncul di jendela pop-up. Anda dapat memilih antara HTTPS, SSH, atau GitHub CLI sesuai dengan preferensi Anda.



Gambar 1: Salin URL Repository

5.2 Membuat Environtment Projek

Langkah kedua adalah membuat environtment projek di local komputer anda. Berikut adalah langkah-langkahnya:

1. Buka aplikasi Code Editor anda, misalnya Visual Studio Code.
2. Tekan tombol `Ctrl + Shift + P` untuk membuka Command Palette.
3. Buat environtment baru dengan memilih versi Python yang sesuai yaitu versi 3.11 kebawah.
4. Kemudian buka terminal dengan menekan tombol `Ctrl + ‘` .
5. Setelah terminal terbuka, ketik perintah berikut untuk meng-clone repository yang sudah anda salin sebelumnya:

   ```
   git clone <URL Repository>
   ```

6. Ganti `<URL Repository>` dengan URL yang sudah anda salin dari GitHub.
7. Tekan `Enter` untuk menjalankan perintah tersebut.
8. Setelah proses cloning selesai, masuk ke direktori projek dengan perintah berikut:

   ```
   cd uas_teknomulmed_shiningshimmeringsplendid
   ```

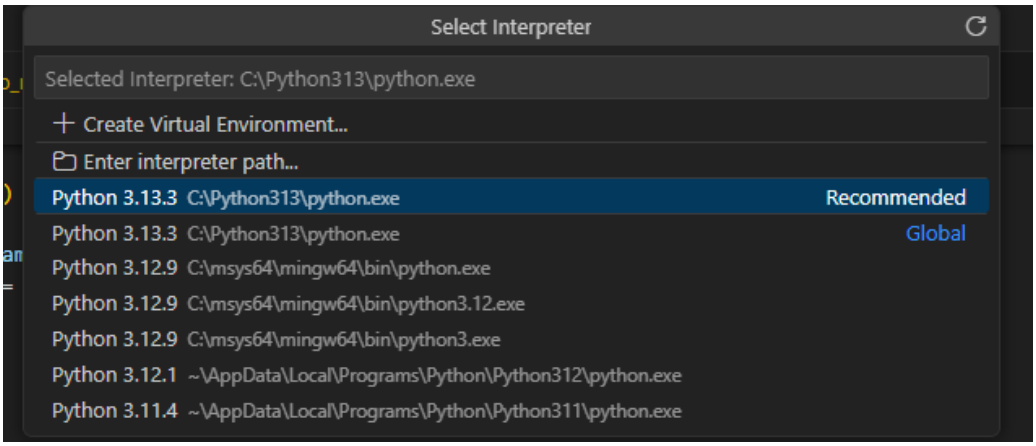
9. Selanjutnya, install semua dependencies yang dibutuhkan oleh projek dengan perintah berikut:

   ```
   pip install -r requirements.txt
   ```

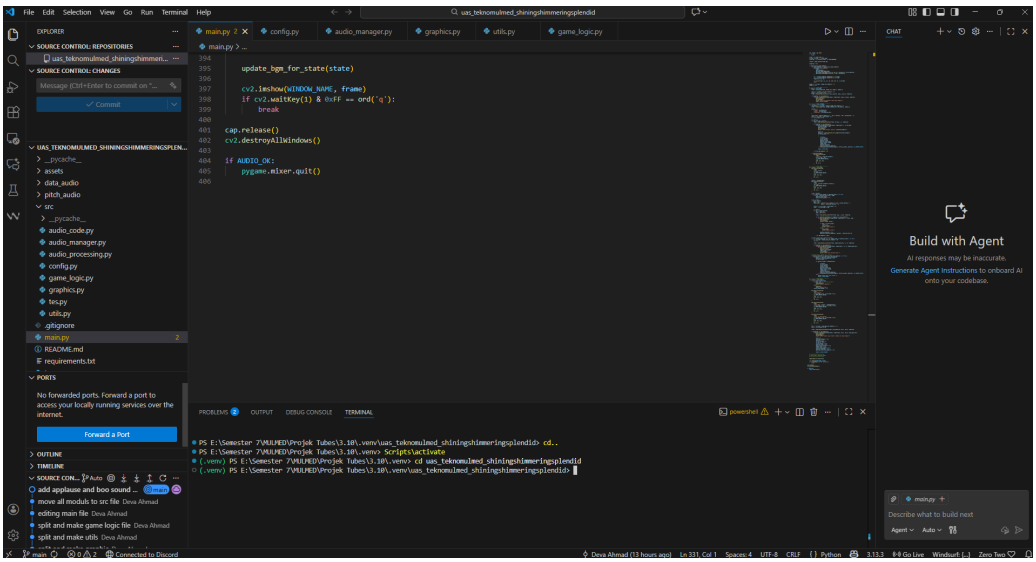
10. Tunggu hingga proses instalasi selesai.
11. Kemudian aktifkan environtment projek dengan perintah berikut:

    ```
    .\venv\Scripts\activate
    ```

12. Sekarang environtment projek sudah aktif dan siap digunakan.



Gambar 2: Memilih Versi Python 3.11 ke bawah



Gambar 3: Tampilan Environtment Projek

5.3 Menjalankan Program

Setelah dependensi terinstall, langkah selanjutnya adalah menjalankan program. Berikut adalah langkah-langkahnya:

1. Pastikan environtment projek sudah aktif.
2. Di terminal, pastikan anda berada di direktori projek.
3. Jalankan program dengan perintah berikut:

```
python main.py
```

4. Tekan `Enter` untuk menjalankan perintah tersebut.
5. Tunggu hingga program berjalan dan jendela aplikasi muncul.
6. Sekarang anda dapat menggunakan aplikasi sesuai dengan fungsinya.

5.4 Tampilan Program Filter

Berikut adalah tampilan dari program filter yang telah dijalankan:

Gambar 4: Tampilan Halaman Utama Program Filter

Gambar 5: Tampilan Halaman Kategori Program Filter

Gambar 6: Tampilan Halaman Permainan Sedang Berlangsung

Gambar 7: Tampilan Halaman Result setelah Permainan selesai

6 Kredit/Sitasi/Referensi

Berikut adalah beberapa referensi yang kami gunakan dalam pembuatan program filter ini:

- Gemini-1
- Gemini-2
- Claude-1
- ChatGPT-1