Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: **Rahmat Aldi Nasda, Shafa Aulia, Naufal Saqib Athaya** Tugas: **Final Project**
Mata Kuliah: **Sistem/Teknologi Multimedia (IF4021)** Tanggal: 26 Mei 2025

1 Deskripsi Project

Final project yang kami buat ini berjudul ***"Swipe That Face!***, sebuah filter interaktif berbasis python yang dikembangkan menggunakan *library MediaPipe*. Tujuan utama dari *final project* ini adalah menciptakan pengalaman interaktif dalam deteksi ekspresi wajah. *Filter* ini dibuat untuk *challenge user* dalam menirukan berbagai ekspresi wajah sesuai dengan emoji yang ditampilkan, khususnya berdasarkan bentuk mata dan bibir. Dalam implementasinya, *user* akan diberikan beberapa level tantangan di mana tiap level menampilkan ekspresi wajah yang berbeda seperti senyum, kaget, sedih, atau ekspresi lainnya. Untuk melanjutkan ke level berikutnya, *user* harus mencocokan ekspresi wajah mereka dengan ekspresi yang ditampilkan.

2 Teknologi

Teknologi yang digunakan dalam pembuatan *final project* ini adalah sebagai berikut :

- ***Python***
  Python merupakan bahasa pemrograman tingkat tinggi, dan serbaguna. *Python* dikenal mudah dalam pengolahan data dan kompatibilitas dengan berbagai *library.* [1, 2].
- ***MediaPipe***
  MediaPipe merupakan *framework* dari *Google* yang digunakan untuk mendeteksi dan melacak ekspresi wajah berdasarkan titik-titik kunci pada wajah secara *real-time* [3]. Fungsinya sangat penting dalam mengenali bentuk mata dan bibir *user.* mediapipe.solutions.face_mesh digunakan untuk mendeteksi 468 titik landmark wajah secara *real-time* [3].
- ***OpenCV (cv2)***
  OpenCV digunakan untuk melakukan proses gambar dan video [4]. Pada project ini *OpenCV* digunakan untuk:
  - Mengakses dan mengambil video dari kamera (cv2.VideoCapture)
  - Menampilkan jendela video (cv2.imshow)
  - Mengubah format warna gambar (cv2.cvtColor)
  - Menggambar teks dan *landmark* wajah pada frame video. (cv2.putText, cv2.line)
- **NumPy**
  NumPy digunakan untuk memproses data numerik dan array yang berkaitan dengan koordinat titik wajah dan perhitungan logika untuk mencocokkan ekspresi [5, 6]. Pada project ini Numpy digunakan untuk :
  - Operasi array saat menggabungkan gambar overlay (emoji) ke frame kamera.

  - Perhitungan dasar seperti rasio dan jarak antar landmark wajah.

- **Tkinter**
  Tkinter adalah modul bawaan *Python* yang digunakan untuk membangun antarmuka pengguna (GUI) [7] seperti tampilan tombol, label level, dan instruksi ekspresi yang harus ditiru oleh user. Dalam project ini Tkinter digunakan untuk :

  - Menampilkan jendela awal yaitu pilihan kamera yang tersedia
  - Menyediakan dropdown dan tombol aksi untuk memulai program.
  - Menyembunyikan jendela utama saat tidak diperlukan.

- ***Webcam***
  Project ini menggunakan kamera device user untuk mengambil video secara langsung melalui fungsi cv2.VideoCapture() dari OpenCV.

- ***FaceMesh Detection Mode*l**
  FaceMesh Detection Model merupakan teknologi yang dibawa oleh *MediaPipe* [3] dimana FaceMesh Detection Model ini digunakan untuk :

  - Mendeteksi wajah
  - Memberikan koordinat pada 468 titik pada wajah, termasuk mata, bibir, dan garis wajah.
  - digunakan dalam fungsi ekspresi wajah seperti senyum, mata tertutup, dan kedipan.

3 Cara Kerja Program

Cara kerja program pada ***Swipe That Face!*** adalah sebagai berikut :

- Pada tahap awal, program menampilkan antarmuka pengguna berbasis Tkinter yang memungkinkan user untuk memilih kamera yang tersedia di devicenya. Antarmuka ini akan menampilkan daftar indeks kamera yang terdeteksi sehingga user bisa memilih kamera yang aktif, lalu user dapat menekan tombol ***"start camera"*** untuk memulai proses deteksi wajah sesuai dengan kamera yang dipilih sebelumnya

- Setelah kamera diaktifkan dan aktif, tiap frame yang diambil dari kamera diproses menggunakan *FaceMesh* dari *MediaPipe. FaceMesh* akan secara otomatis mendeteksi titik *landmark* di wajah seperti mata, hidung, bibir, dan kontur wajah. Landmark akan terus diperbaharui tergantung pergerakan wajah *user*.

- Berdasarkan posisi tiap titik yang diperoleh oleh *FaceMesh*, program akan menghitung jarak antar titik tertentu pada wajah untuk mendeteksi ekspresi. Jika jarak antara bibir melebihi parameter tertentu maka akan dianggap tersenyum. Jika salah satu mata terdeteksi tertutup berdasarkan jarak lebar tinggi mata maka akan dianggap berkedip dan jika kedua mata tertutup secara bersamaan maka dianggap menutup mata secara keseluruhan.

- Tiap ekspresi yang terdeteksi oleh frame akan dikirimkan ke modul permainan yang telah dibuat di dalam file expression.py. Modul ini akan menjadi pengendali logika pada program ***Swipe That Face!*** ini. Misalkan permainan dapat meminta pengguna untuk menirukan ekspresi sesuai emoji dan akan mencatat apakah ekspresinya sesuai dengan yang diminta.

- Sebagai visual tambahan, program menampilkan emoji di atas kiri posisi wajah user. Emoji ditambahkan menggunakan fungsi *overlay* dan informasi terkait permainan seperti ekspresi yang diminta, skor, atau umpan balik di gambar langsung di frame menggunakan *library OpenCV.*

- Frame yang telah diproses secara lengkap kemudian ditampilkan ke user melalui jendela *OpenCV* dan akan terus diperbaharui seiring dengan proses deteksinya.
- Jika pengguna menekan tombol 'q' atau ESC, maka program akan berhenti memproses video, menutup *webcam*, dan menutup semua jendela yang terbuka.

4 Penjelasan Kode Program

Kode program untuk project ***Swipe That Face!*** ini dibagi menjadi beberapa bagian kode yaitu **expression.py**, **facedetect.py**, **game.py**, **gui.py**, **main.py**, **qtgame.py**. Berikut merupakan penjelasan dari masing-masing bagian kode program:

4.1 expression.py

File **expression.py** merupakan file yang isinya membentuk pondasi sistem deteksi ekspresi wajah user melalui kamera. Setiap fungsi fokus pada deteksi bagian wajah tertentu dengan pendekatan geometris berdasarkan *landmark* wajah dari *MediaPipe*. Program ini memanfaatkan library OpenCv, NumPy, dan MediaPipe untuk mengenali ekspresi seperti senyum, senyum lebar, kedipan, dan mata tertutup yang juga bisa memberikan umpan balik visual berupa sistem permainan ekspresi.

```
import cv2
import numpy as np
import tkinter as tk
from facedetect import FaceDetector

tk.Tk().withdraw()  # Hide main tkinter window for popup use

def is_smiling(landmarks, image_width, image_height):
    """Cek apakah user tersenyum berdasarkan FaceMesh landmark."""

    # Ambil titik-titik yang relevan
    def get_coord(idx):
        pt = landmarks.landmark[idx]
        return int(pt.x * image_width), int(pt.y * image_height)

    # Titik penting
    left_corner = get_coord(61)
    right_corner = get_coord(291)
    mid_upper_lip = get_coord(13)
    bottom_lip = get_coord(14)

    # Mulut harus tertutup (opsional tapi penting)
    mouth_open = abs(bottom_lip[1] - mid_upper_lip[1])
    is_mouth_closed = mouth_open < 5

    # Senyum: sudut bibir lebih tinggi dari titik tengah
    left_is_higher = left_corner[1] < mid_upper_lip[1] - 3
    right_is_higher = right_corner[1] < mid_upper_lip[1] - 3
    is_corner_up = left_is_higher and right_is_higher

    return is_mouth_closed and is_corner_up

def is_big_smiling(landmarks, image_width, image_height):
    """Detect big smile by measuring vertical lip distance ratio"""
    def get_coord(idx):
        pt = landmarks.landmark[idx]
        return int(pt.x * image_width), int(pt.y * image_height)

    left_mouth, right_mouth = get_coord(61), get_coord(291)
```

```
    top_lip, bottom_lip = get_coord(13), get_coord(14)

    mouth_width = abs(right_mouth[0] - left_mouth[0])
    mouth_height = abs(bottom_lip[1] - top_lip[1])

    smile_ratio = mouth_width / (mouth_height + 1e-5)

    # Big smile = mulut terbuka cukup tinggi, tapi tetap lebar
    return smile_ratio > 1.8 and mouth_height > 20

def is_blinking(landmarks, image_width, image_height):
    """Check if user is blinking (either left or right eye)"""
    def get_coord(idx):
        pt = landmarks.landmark[idx]
        return int(pt.x * image_width), int(pt.y * image_height)

    # Get eye landmarks
    left_eye_top = get_coord(159)
    left_eye_bottom = get_coord(145)
    right_eye_top = get_coord(386)
    right_eye_bottom = get_coord(374)

    # Calculate eye aspect ratios
    left_eye_height = abs(left_eye_top[1] - left_eye_bottom[1])
    right_eye_height = abs(right_eye_top[1] - right_eye_bottom[1])

    # Check if either eye is closed (small height indicates closed eye)
    threshold = 5
    return left_eye_height < threshold or right_eye_height < threshold

def is_eyes_closed(landmarks, image_width, image_height):
    """Check if both eyes are closed"""
    def get_coord(idx):
        pt = landmarks.landmark[idx]
        return int(pt.x * image_width), int(pt.y * image_height)

    # Get eye landmarks for both eyes
    left_eye_top = get_coord(159)
    left_eye_bottom = get_coord(145)
    right_eye_top = get_coord(386)
    right_eye_bottom = get_coord(374)

    # Calculate eye heights
    left_eye_height = abs(left_eye_top[1] - left_eye_bottom[1])
    right_eye_height = abs(right_eye_top[1] - right_eye_bottom[1])

    # Mata (booth) harus nutup
    threshold = 5
    return left_eye_height < threshold and right_eye_height < threshold

def overlay_img(background, overlay, pos):
    """Menggabungkan overlay (dengan transparansi) ke frame utama."""
    x, y = pos
    h, w = overlay.shape[:2]

    # Cek apakah ada alpha channel
    if overlay.shape[2] == 4:
        alpha = overlay[:, :, 3] / 255.0
        for c in range(3):
            background[y:y+h, x:x+w, c] = (
                alpha * overlay[:, :, c] +
                (1 - alpha) * background[y:y+h, x:x+w, c]
```

```
            )
    else:
        background[y:y+h, x:x+w] = overlay

def show_expression_game(cam_index: int, emoji_path: str):
    from PyQt6.QtWidgets import QApplication
    from qtgame import GameWindow
    import sys

    app = QApplication(sys.argv)
    window = GameWindow(cam_index)
    window.show()
    sys.exit(app.exec())
```

4.1.1 Fungsi `is_smilling()`

- Mendeteksi senyum dengan melakukan pemeriksaan pada posisi *landmark* bibir.
- Mengecek apakah bibir tertutup dan apakah sudut bibir lebih tinggi dari bibir tengah yang menandakan ekspresi senyum.

4.1.2 Fungsi `is_big_smilling()`

- Mendeteksi senyum lebar *(big smile)* dengan mendeteksi rasio lebar bibir terhadap tinggi bibir.
- Digunakan pula untuk mengenali ekspresi tertawa atau tersenyum lebar dengan bibir terbuka.

4.1.3 Fungsi `is_blinking()`

- Mendeteksi kedipan mata dengan melihata apakah salah satu mata mengalami penurunan vertikal diantara titik atas bawah mata.
- jika ukuran mata sangat sempit yaitu <5 pixel maka akan dianggap berkedip.

4.1.4 Fungsi `is_eyes_closed()`

- Hampir sama dengan fungsi sebelumya namun fungsi ini berfungsi untuk memeriksa apakah kedua mata user tertutup.

4.1.5 Fungsi `overlay_img()`

- Digunakan untuk menampilkan gambar emoji diatas frame webcam secara presisi.

4.1.6 Fungsi `show_expression_game()`

- Inisiasi untuk menjalankan game **Show That Face!**

4.2 facedetect.py

File **facedetect.py** merupakan file yang berfungsi untuk mendeteksi dan menggambar l*andmark* wajah menggunakan *MediaPipe* dan *OpenCv*. File ini memiliki dua fungsi utama yaitu mendeteksi *landmark* wajah dari frame video dan menampilkan visualisasi *landmark* pada frame. Kelas ini memungkinkan sistem untuk mendeteksi ekspresi wajah dengan mendapatkan titik-titik wajah yang presisi dan memvisualisasikannya pada frame video webcam.

```python
import mediapipe as mp
import cv2

class FaceDetector:
    def __init__(self, max_faces=1):
        self.mp_face_mesh = mp.solutions.face_mesh
        self.mp_drawing = mp.solutions.drawing_utils
        self.face_mesh = self.mp_face_mesh.FaceMesh(
            static_image_mode=False,
            max_num_faces=max_faces,
            refine_landmarks=True,
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5
        )

    def get_landmarks(self, frame):
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = self.face_mesh.process(rgb_frame)
        return results.multi_face_landmarks

    def draw_landmarks(self, frame, landmarks):
        # Draw title text
        ih, iw, _ = frame.shape
        title = "Swipe That Face!"
        font = cv2.FONT_HERSHEY_DUPLEX  # Closest to Poppins available in OpenCV
        font_scale = 1.2
        thickness = 2
        text_size = cv2.getTextSize(title, font, font_scale, thickness)[0]
        text_x = (iw - text_size[0]) // 2  # Center horizontally
        text_y = 50  # Position from top

        # Draw text shadow/outline for better visibility
        cv2.putText(frame, title, (text_x+2, text_y+2), font, font_scale, (0, 0, 0), thickness+1)
        cv2.putText(frame, title, (text_x, text_y), font, font_scale, (0, 255, 0), thickness)

        # Draw landmarks
        for face_landmarks in landmarks:
            self.mp_drawing.draw_landmarks(
                image=frame,
                landmark_list=face_landmarks,
                connections=self.mp_face_mesh.FACEMESH_TESSELATION,
                landmark_drawing_spec=None,
                connection_drawing_spec=self.mp_drawing.DrawingSpec(
                    color=(0, 255, 0),
                    thickness=1,
                    circle_radius=1
                )
            )
        return frame  # Return the modified frame
```

4.2.1 Inisialisasi Kelas *FaceDetector*

- Konstruktor __init__() menerima parameter max_faces (jumlah max wajah yang dapat di deteksi, def:1)
- Inisialisasi model *FaceMesh* dari *MediaPipe* untuk deteksi titik-titik *landmark* wajah dengan beberapa konfigurasi seperti :
  - static_image_mode = false yang artinya input berupa video bukan gambar.
  - max_num_faces=max_faces yaitu jumlah max wajah yang akan dianalisis.

- refine_landmarks=True yaitu untuk mendeteksi *landmark* lebih detail (iris mata).
- min_detection_confidence dan min_tracking_confidence untuk batas minimun validasi deteksi dan pelacakan.

4.2.2 Fungsi get_landmarks(frame)

- Mengubah warna dari frame BGR (*OpenCV*) ke RGB (*MediaPipe*).
- Menjalankan model *FaceMesh* untuk memproses wajah dan mengembalikan koordinat *landmark* wajah dari frame tersebut.

4.2.3 Fungsi draw_landmark(frame, landmarks)

- Pertama-tama menggambar judul "***Swipe That Face!***" diatas frame, dengan bayangan agar terlihat lebih jelas.
- Judul ditampilan pada bagian atas tengah layar menggunakan fungsi font dan berwarna hijau, kemudian menggambar landmark wajah dari *MediaPipe*
- Setiap *landmark* wajah digambar dengan warna hijau dan lingkaran kecil sebagai penanda titik.

4.3 game.py

Kode program ini berisi implementasi dari sebuah permainan berbasis *webcam* dan deteksi ekspresi wajah. Tujuan dari game ini adalah mendeteksi ekspresi wajah tertentu seperti senyum, senyum, lebar, berkedip, dan mata tertutup) menggunakan model deteksi wajah lalu menampilkan emoji ekspresi yang harus diikuti user. Saat user berhasil menirukan ekspresi yang diminta akan muncul suara menang.

```
import cv2
import numpy as np
from facedetect import FaceDetector
from expression import is_smiling, is_big_smiling, is_blinking, is_eyes_closed
from playsound import playsound
import threading
import sys

class ExpressionGame:
    def __init__(self, cam_index):
        self.cap = cv2.VideoCapture(cam_index)
        # Set camera properties to 1280x720
        self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
        self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
        self.detector = FaceDetector()
        self.window_size = (1280, 720)  # Changed from (640, 640)
        self.setup_window = lambda: None  # Disable CV2 window creation
        self.setup_game_states()
        self.load_expressions()
        self.success_sound = "sound/berhasil.mp3"
        self.show_landmarks = True  # Add this line to track landmark visibility
        self.audio_threads = []  # Add this line to track audio threads

    def setup_game_states(self):
        self.MENU = 0
        self.COUNTDOWN = 1
        self.PLAYING = 2
        self.game_state = self.MENU
        self.countdown_start = None
        self.countdown_duration = 1.0
        self.win_start_time = None
```

```
        self.WIN_DURATION = 3
        self.game_completed = False
        self.current_expression = 0

    def load_expressions(self):
        self.expressions = [
            {"emoji": "Assets/senyum.png", "detector": is_smiling},
            {"emoji": "Assets/senyumlebar.png", "detector": is_big_smiling},
            {"emoji": "Assets/kedip.png", "detector": is_blinking},
            {"emoji": "Assets/merem.png", "detector": is_eyes_closed},
        ]
        self.current_emoji = cv2.imread(self.expressions[0]["emoji"], cv2.IMREAD_UNCHANGED)
        if self.current_emoji is None:
            raise FileNotFoundError("Emoji image not found!")
        self.current_emoji = cv2.resize(self.current_emoji, (150, 150))

    def reset_game(self):
        self.current_expression = 0
        self.win_start_time = None
        self.game_completed = False
        self.current_emoji = cv2.imread(self.expressions[0]["emoji"], cv2.IMREAD_UNCHANGED)
        self.current_emoji = cv2.resize(self.current_emoji, (150, 150))

    def draw_ui(self, frame, text, position, font_scale=1.2, color=(0, 255, 0)):
        h, w = frame.shape[:2]
        font = cv2.FONT_HERSHEY_DUPLEX
        thickness = 2
        text_size = cv2.getTextSize(text, font, font_scale, thickness)[0]

        if position == 'center':
            text_x = (w - text_size[0]) // 2
            text_y = (h + text_size[1]) // 2
        elif position == 'top':
            text_x = (w - text_size[0]) // 2
            text_y = 50
        elif position == 'bottom':
            text_x = (w - text_size[0]) // 2
            text_y = h - 50

        cv2.putText(frame, text, (text_x+2, text_y+2), font, font_scale, (0, 0, 0), thickness+1)
        cv2.putText(frame, text, (text_x, text_y), font, font_scale, color, thickness)

    def show_main_menu(self, frame):
        self.draw_ui(frame, "Swipe That Face!", 'top', 1.5, (0, 255, 0))
        # Removed "Press SPACE to PLAY" text

    def show_countdown(self, frame, number):
        self.draw_ui(frame, str(number), 'center', 4.0, (0, 255, 255))

    def overlay_emoji(self, frame, emoji, pos):
        x, y = pos
        h, w = emoji.shape[:2]
        if emoji.shape[2] == 4:
            alpha = emoji[:, :, 3] / 255.0
            for c in range(3):
                frame[y:y+h, x:x+w, c] = (
                    alpha * emoji[:, :, c] +
                    (1 - alpha) * frame[y:y+h, x:x+w, c]
                )
        else:
            frame[y:y+h, x:x+w] = emoji

```

```
    def cleanup(self):
        """Cleanup resources before exit"""
        if self.cap is not None:
            self.cap.release()
        cv2.destroyAllWindows()

        # Wait for any audio threads to complete
        for thread in self.audio_threads:
            if thread.is_alive():
                thread.join(timeout=1.0)

    def play_success_sound(self):
        """Play sound in thread and track it"""
        thread = threading.Thread(
            target=playsound,
            args=(self.success_sound,),
            daemon=True
        )
        self.audio_threads.append(thread)
        thread.start()

    def get_current_frame(self):
        """Get processed frame for current game state"""
        success, frame = self.cap.read()
        if not success:
            return None

        # Process frame
        frame = self.process_frame(frame)

        # Handle different game states
        if self.game_state == self.MENU:
            self.show_main_menu(frame)
        elif self.game_state == self.COUNTDOWN:
            self.handle_countdown(frame)
        elif self.game_state == self.PLAYING:
            self.handle_gameplay(frame)

        return frame

    def play_success_sound(self):
        # Play in separate thread to avoid blocking
        threading.Thread(target=playsound, args=(self.success_sound,), daemon=True).start()

    def process_frame(self, frame):
        """Process the frame for gameplay"""
        h, w = frame.shape[:2]
        target_w, target_h = self.window_size

        # Calculate scaling factor
        scale = min(target_w/w, target_h/h)
        new_w = int(w * scale)
        new_h = int(h * scale)

        # Resize frame preserving aspect ratio
        frame = cv2.resize(frame, (new_w, new_h))

        # Create black canvas of target size
        canvas = np.zeros((target_h, target_w, 3), dtype=np.uint8)

        # Center the frame on canvas
        y_offset = (target_h - new_h) // 2
```

```
        x_offset = (target_w - new_w) // 2
        canvas[y_offset:y_offset+new_h, x_offset:x_offset+new_w] = frame

        return canvas

    def start_game(self):
        """Start game from menu state"""
        if self.game_state == self.MENU:
            self.game_state = self.COUNTDOWN
            self.countdown_start = cv2.getTickCount()

    def handle_gameplay(self, frame):
        h, w = frame.shape[:2]
        landmarks_list = self.detector.get_landmarks(frame)

        self.draw_ui(frame, "Swipe That Face!", 'top')

        if landmarks_list:
            if self.show_landmarks:
                self.detector.draw_landmarks(frame, landmarks_list)

            if not self.game_completed:
                expression_detected = self.expressions[self.current_expression]["detector"](
                    landmarks_list[0], w, h)
                if expression_detected and self.win_start_time is None:
                    self.win_start_time = cv2.getTickCount()

        self.handle_win_state(frame)

        if not self.game_completed:
            self.overlay_emoji(frame, self.current_emoji, (10, 10))
        else:
            # Changed message to be simpler
            self.draw_ui(frame, "Game Completed!", position='center', font_scale=1.0, color=(255,
    255, 255))

    def handle_countdown(self, frame):
        """Handle countdown state"""
        current_time = cv2.getTickCount()
        elapsed = (current_time - self.countdown_start) / cv2.getTickFrequency()
        countdown_number = 3 - int(elapsed)

        if countdown_number > 0:
            self.show_countdown(frame, countdown_number)
        else:
            self.game_state = self.PLAYING
            self.reset_game()

    def handle_win_state(self, frame):
        """Handle win state logic"""
        if self.win_start_time is not None and not self.game_completed:
            current_time = cv2.getTickCount()
            elapsed = (current_time - self.win_start_time) / cv2.getTickFrequency()

            if elapsed <= self.WIN_DURATION:
                self.draw_ui(frame, "Kamu berhasil!", 'center')
                # Play sound when first entering win state
                if elapsed < 0.1:  # Only play once at the start of win state
                    self.play_success_sound()
            else:
                if self.current_expression < len(self.expressions) - 1:
                    self.current_expression += 1
```

```
                    self.current_emoji = cv2.imread(
                        self.expressions[self.current_expression]["emoji"],
                        cv2.IMREAD_UNCHANGED
                    )
                    self.current_emoji = cv2.resize(self.current_emoji, (150, 150))
                else:
                    self.game_completed = True
                self.win_start_time = None
```

4.3.1 Class ExpressionGame

Merupakan inti dari program game ekspresi wajah ini; mengatur seluruh alur permainan, menu, countdown, game play, hingga kemenangan.

4.3.2 Fungsi def __init__(self, cam_index)

- Inisialisasi ovjek game
- Membuka kamera dengan index yang dipilih
- Mengatur resolusi kamera
- Membuat objek pendeteksi wajah (FaceDetector) dan mengatur ukuran window.
- Membuat emoji dan ekspresi yang akan digunakan selama game berjalan.
- Mengatur state awal game, suara, dan tampilan landmark.

4.3.3 Fungsi def setup_game_states(self):

- Mengatur konstanta dan variabek terkait state game; menu, countdown, playing, dll.
- Menyimpan waktu mulai countdown dan kemenangan

4.3.4 Fungsi def load_expression(self)

- Membuat daftar ekspresi emoji
- Menghubungkan ekspresi dengan detector fungsional dari moudul expression.
- Memuat dan mengatur ukuran emoji yang ditampilkan

4.3.5 Fungsi def reset_game(self):

- Mengulang ekspresi ke awal dan mengatur ulang status kemenangan.

4.3.6 Fungsi def show_main_menu(self, frame)

- Menampilkan judul game saat memulai state Menu.

4.3.7 Fungsi def show_countdown(self, frame, number)

- Menampilkan angka countdown di layar.

4.3.8 Fungsi def overlay_emoji(self, frame, emoji, pos)

- Menambahkan emoji ke dalam frame dalam posisi tertentu.
- Mengelola transparansi gambar emoji.

4.3.9 Fungsi def cleanup(self)

- Menutup kamera dan semua jendela

4.3.10 Fungsi def play_success_sound(self)

- Memainkan sound keberhasilan dalam thread terpisah agar tidak memblokir alur program

4.3.11 Fungsi def get_get_current_frame(self))

- Membaca dan memproses frame dari kamera
- menjalankan logika state game

4.3.12 Fungsi def process_frame(self,frame)

- Mengubah ukuran frame agar sesuai

4.3.13 Fungsi def start_game(self)

- Menguba state game dari menu ke countdown dan mencatat waktu mulai.

4.3.14 Fungsi def handle_gameplay(self,frame):

- Menangani logika permainan; Deteksi landmark wajah, pengecekan keberhasilan ekspresi yang dilakukan, tampilkan emoji, deteksi kemenangan untuk tiap ekspresi.

4.3.15 Fungsi def handle_countdown(self,frame):

- Mengatur transisi dari countdown ke playing dan menampilkan angka countdown.

4.3.16 Fungsi def handle_win_state(self, frame):

- Ketika ekspresi berhasil dideteksi dan sesuai dengan ekspresi yang diminta maka akan muncul tulisan “Kamu berhasil!”.
- Suara kemenangan diputar menggunakan *playsound* di thread terpisah.
- Jika ekspresi masih ada yang belum dikerjakan, maka berlanjut ke ekspresi berikutnya.

4.4 gui.py

Kode pada file gui.py merupakan modul yang menggunakan *library* TKinter yang berfungsi untuk membuat tampilan antarmuka program. Kode dalam file ini terintegrasi dengan sistem deteksi wajah dan ekspresi melalui modul lain yaitu **facedetect.py**.

```
import tkinter as tk
from tkinter import ttk, PhotoImage
import cv2
from facedetect import FaceDetector
from utils import list_available_cameras
from expression import show_expression_game
from PyQt6.QtGui import QIcon

def run_landmark_view(cam_index):
    cap = cv2.VideoCapture(cam_index)
    detector = FaceDetector()

    while cap.isOpened():
        success, frame = cap.read()
        if not success:
            print("Gagal ambil frame.")
            break

        landmarks = detector.get_landmarks(frame)
        if landmarks:
            detector.draw_landmarks(frame, landmarks)

        cv2.imshow(f'Face Landmark Viewer - Camera {cam_index}', frame)

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

    cap.release()
    cv2.destroyAllWindows()

def get_available_cameras():
    """Deteksi kamera yang benar-benar tersedia"""
    available_cameras = []
    max_test = 10  # Test sampai index 10

    for i in range(max_test):
        cap = cv2.VideoCapture(i, cv2.CAP_DSHOW)  # Gunakan DirectShow
        if cap.isOpened():
            # Ambil informasi kamera
            ret, frame = cap.read()
            if ret:
                # Dapatkan nama device
                name = f"Camera {i}"
                try:
                    # Coba dapatkan nama backend
                    backend = cap.getBackendName()
                    if backend:
                        name = f"{backend} Camera {i}"
                except:
                    pass
                available_cameras.append((i, name))
            cap.release()
    return available_cameras

class CameraSelector(tk.Toplevel):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.selected_camera = None
        self.camera_indices = []
        # Change icon setting method
        self.icon = PhotoImage(file='logo/icon.png')
        self.tk.call('wm', 'iconphoto', self._w, self.icon)
```

```
        self.initUI()
        self.center_window()

    def center_window(self):
        # Get screen width and height
        screen_width = self.winfo_screenwidth()
        screen_height = self.winfo_screenheight()

        # Calculate window position
        x = (screen_width - 800) // 2  # 800 is window width
        y = (screen_height - 600) // 2  # 600 is window height

        # Set window position
        self.geometry(f'800x600+{x}+{y}')

    def initUI(self):
        self.title('Select Camera')
        self.geometry('800x600')  # Ukuran window lebih besar
        self.resizable(False, False)

        # Style untuk judul
        title_label = tk.Label(
            self,
            text="Select Your Camera",
            font=('Arial', 24, 'bold'),  # Font lebih besar
            pady=20
        )
        title_label.pack(pady=40)  # Spacing lebih besar

        # Detect available cameras
        cameras = get_available_cameras()

        if not cameras:
            label = tk.Label(
                self,
                text="No cameras found!",
                font=('Arial', 16),  # Font lebih besar
                fg='red'
            )
            label.pack(pady=30)
        else:
            # Store camera indices and names
            self.camera_indices = [idx for idx, name in cameras]
            camera_names = [name for idx, name in cameras]

            # Style combobox
            style = ttk.Style()
            style.configure('Custom.TCombobox', font=('Arial', 14))  # Font lebih besar

            self.combo = ttk.Combobox(
                self,
                values=camera_names,
                state='readonly',
                width=40,  # Width lebih besar
                font=('Arial', 14),  # Font lebih besar
                style='Custom.TCombobox'
            )
            self.combo.pack(pady=(30, 20))  # Spacing lebih besar
            self.combo.current(0)

            # Style button
            select_btn = ttk.Button(
```

```
                self,
                text='Start Camera',  # Text lebih deskriptif
                command=self.on_select,
                style='Custom.TButton'
            )

            # Custom style untuk button
            style.configure(
                'Custom.TButton',
                font=('Arial', 14, 'bold'),  # Font lebih besar
                padding=10
            )

            select_btn.pack(pady=30)  # Spacing lebih besar

    def on_select(self):
        current_idx = self.combo.current()  # Use current() instead of currentIndex()
        self.selected_camera = self.camera_indices[current_idx]  # Get the actual camera index
        self.destroy()

def launch_camera_selector():
    dialog = CameraSelector()
    dialog.grab_set()  # Make the dialog modal
    dialog.wait_window()  # Wait for the dialog to close
    return dialog.selected_camera if dialog.selected_camera is not None else -1

def main():
    cam_index = launch_camera_selector()
    if cam_index != -1:
        run_landmark_view(cam_index)

if __name__ == "__main__":
    root = tk.Tk()
    root.withdraw()  # Hide the root window
    main()
```

4.4.1 Fungsi def run_landmark_view(cam_index):

- Fungsi ini digunakan untuk melihat *landmark* wajah secara langsung melalui kamera. Wajah akan dideteksi dan *landmark* (titik-titik penting wajah) digambar pada frame video secara *real-time*. Fitur ini bersifat tambahan untuk *debugging atau visualisasi.*

4.4.2 def get_available_cameras()

- Melakukan pengecekan pada kamera yang tersedia sesuai index.
- Jika kamera aktif dan dapat membaca frame maka index akan masuk ke dalam list.
- Mengembalikan daftar tuple (index, nama kamera)

4.4.3 Class CameraSelector(tk.Toplevel)

Kelas untuk membuat dialog pemilihan kamera.

4.4.4 def launch_camera_selector():

- Menjalankan CameraSelector dengan membuat jendela utama tidak aktif sampai dialog selesai
- Menunggu pengguna memilih kamera dan menutup dialog.

4.4.5 main()

- Memanggil fungsi launch_camera_selector() untuk mendapatkan indeks kamera.
- lalu akan menjalankan run_landmark_view().

4.4.6 Blok if__name__=="__main__":

- Membuat jendela Tkinter utama namun menyembunyikannya
- Menjalankan fungsi utama program

4.5 main.py

File **main.py** merupakan file utama yang digunakan untuk menjalankan aplikasi ***Swipe That Face!***. File merupakan titik awal eksekusi program yang akan memanggil fungsi antarmuka pemilihan kamera yang ada di file gui.py agar user bisa memulai game.

4.5.1 Import Fungsi launch_camera_selector dari Modul gui

from gui import launch_camera_selector
Line ini mengimpor fungsi launch_camera_selector dari file gui.py. fungsi ini akan menampilkan antarmuka pengguna yang memungkinkan user untuk memilih kamera dan mulai untuk melakukan proses deteksi ekspresi wajah.

4.5.2 Blok Kondisional if __name__ == "__main__"

Line ini memastikan bahwa fungsi launch_camera_selector() akan berjalan hanya jika file main.py dieksekusi secara langsung oleh Python bukan diimport dari file lain.

4.6 qtgame.py

File **qtgame.py** berisikan tentang komponen antarmuka pengguna (GUI) dari projek yang kami kembangkan **"Swipe That Face!"** yang dibangun menggunakan **PyQt6**.

```
from PyQt6.QtWidgets import (QMainWindow, QWidget, QVBoxLayout,
                             QHBoxLayout, QPushButton, QLabel, QFrame)
from PyQt6.QtCore import Qt, QTimer
from PyQt6.QtGui import QImage, QPixmap, QIcon
import cv2
import threading
from playsound import playsound
import numpy as np
from game import ExpressionGame

class GameWindow(QMainWindow):
    def __init__(self, cam_index):
        super().__init__()
        self.game = ExpressionGame(cam_index)
        self.is_closing = False
        # Start background music in a separate thread
        self.bg_music_thread = threading.Thread(
            target=self.play_background_music,
            daemon=True
        )
        self.bg_music_thread.start()
        self.setWindowIcon(QIcon('logo/icon.png'))
        self.initUI()

```

```
    def play_background_music(self):
        while not self.is_closing:
            playsound("sound/backsound.mp3")

    def closeEvent(self, event):
        self.is_closing = True
        if hasattr(self, 'timer'):
            self.timer.stop()
        self.game.cleanup()
        event.accept()

    def initUI(self):
        # Main widget and layout
        central_widget = QWidget()
        self.setCentralWidget(central_widget)
        main_layout = QVBoxLayout(central_widget)
        main_layout.setSpacing(10)  # Add space between containers

        # Video container
        video_container = QFrame()
        video_container.setFrameStyle(QFrame.Shape.Box)
        video_container.setStyleSheet("QFrame { background-color: black; }")
        video_layout = QVBoxLayout(video_container)

        # Video feed label
        self.video_label = QLabel()
        self.video_label.setMinimumSize(1280, 720)
        self.video_label.setAlignment(Qt.AlignmentFlag.AlignCenter)
        video_layout.addWidget(self.video_label)

        # Add video container to main layout
        main_layout.addWidget(video_container)

        # Button container
        button_container = QFrame()
        button_container.setFixedHeight(80)  # Set fixed height for button area
        button_container.setStyleSheet("""
            QFrame {
                background-color: #34495e;
                border-radius: 10px;
                padding: 10px;
            }
        """)

        # Button layout
        button_layout = QHBoxLayout(button_container)
        button_layout.setSpacing(20)  # Space between buttons
        button_layout.setContentsMargins(20, 0, 20, 0)

        # Create and store buttons as instance variables
        self.start_btn = QPushButton("Start Game")
        self.landmark_btn = QPushButton("Toggle Landmarks")
        self.reset_btn = QPushButton("Reset Game")
        self.quit_btn = QPushButton("Quit")

        # Store buttons in a list for easy access
        self.buttons = [self.start_btn, self.landmark_btn, self.reset_btn, self.quit_btn]

        button_style = """
            QPushButton {
                background-color: #2ecc71;
                color: white;
```

```
                border: none;
                border-radius: 5px;
                padding: 10px 20px;
                font-weight: bold;
                font-size: 14px;
                min-width: 150px;
            }
            QPushButton:hover {
                background-color: #27ae60;
            }
            QPushButton:pressed {
                background-color: #1e8449;
            }
            QPushButton:disabled {
                background-color: #95a5a6;
            }
        """

        # Add all buttons to layout
        for btn in self.buttons:
            btn.setStyleSheet(button_style)
            button_layout.addWidget(btn)

        # Add button container to main layout
        main_layout.addWidget(button_container)

        # Connect buttons
        self.start_btn.clicked.connect(self.start_game)
        self.landmark_btn.clicked.connect(self.toggle_landmarks)
        self.reset_btn.clicked.connect(self.reset_game)
        self.quit_btn.clicked.connect(self.close)

        # Initial button states
        self.reset_btn.setEnabled(False)

        # Window setup
        self.setWindowTitle('Swipe That Face!')
        self.setMinimumSize(1280, 820)  # 720 + button area + margins
        self.setStyleSheet("background-color: #2c3e50;")  # Dark background

        # Setup timer for video update
        self.timer = QTimer()
        self.timer.timeout.connect(self.update_frame)
        self.timer.start(30)  # 30ms = ~33fps

    def start_game(self):
        """Handle start button click"""
        self.game.start_game()
        self.start_btn.setEnabled(False)
        self.reset_btn.setEnabled(False)

    def update_frame(self):
        if self.is_closing:
            return

        frame = self.game.get_current_frame()
        if frame is not None:
            h, w = frame.shape[:2]
            q_img = QImage(frame.data, w, h, frame.strides[0], QImage.Format.Format_BGR888)
            self.video_label.setPixmap(QPixmap.fromImage(q_img))

            # Update button states only if window is still active
```

```
            try:
                if self.game.game_completed:
                    self.reset_btn.setEnabled(True)
                    self.start_btn.setEnabled(False)
                elif self.game.game_state == self.game.MENU:
                    self.start_btn.setEnabled(True)
                    self.reset_btn.setEnabled(False)
            except RuntimeError:
                # Handle case where buttons might be deleted
                pass

    def toggle_landmarks(self):
        self.game.show_landmarks = not self.game.show_landmarks

    def reset_game(self):
        """Handle reset button click"""
        if self.game.game_completed:
            self.game.reset_game()
            self.game.game_state = self.game.MENU
            self.start_btn.setEnabled(True)
            self.reset_btn.setEnabled(False)
```

4.6.1 Kelas GameWindow

class GameWindow(QMainWindow):
Merupakan kelas utama GUI yang menurunkan QMAinWindow, Konstruktor :

4.6.2 Fungsi def __init__(self, cam_index)

Inisialisasi window dan objek game (ExspressionGame(cam_index)), memulai musik latar belakang dalam thread terpisah, memuat ikon, dan memanggil initUI() untuk membangun UI.

4.6.3 Fungsi def play_background_music(self)

Memutar file musik secara loop dan berjalan di thread daemon sampai jendela tertutup.

4.6.4 closeEvent(self, event)

Mengatur is_closing menjadi True, menghentikan timer, dan membersihkan sumber daya dari game saat jendela closed.

4.6.5 Fungsi initUI(self)

Membangun semua elemen UI

- Membuat layout utama berbentuk vertikal (QVBoxLayout).
- Membuat video container menggunakan label untuk menampilkan video webcam.
- Membuat button container empat tombol
  - Start game
  - Toggle Landmarks
  - Reset Game
  - Quit
- Mengatur gaya tombol dengan CSS Qt.
- Mengatur timer untuk memanggil update_frame().

4.6.6 Fungsi start_game(self)

Dipanggil ketika tombol start game ditekan

4.6.7 Fungsi update_frame(self)

Dipanggil secara berkala oleh QTimer

- Mengambil frame dari kamera (self.game.get_current_frame()).
- Mengubah formatnya menjadi QImage lalu ditampilkan di QLabel.
- Memeriksa status game.

4.6.8 Fungsi toggle_landmarks(self)

Membalik status boolean self.game.show_landmarks yang digunakan untuk menyembunyikan/menampilan titik-titik landmark di wajah.

4.6.9 Fungsi reset_game(self)

- Memanggil reset_game() untuk mengulang permainan.
- Mengatur ulang status tombol

5 Deskripsi Library

Berikut adalah deskripsi dari tiap *library* yang digunakan dalam keseluruhan program pada final project ***Swipe That Face!***

5.1 CV2 (OpenCV)

Library ini digunakan untuk memeroses gambar dan video, fungsinya mencakup membuka dan mengambil video dari camera (VideoCapture). OpenCV menjadi bagian yang penting dalam pemrosesan visual untuk deteksi dan menampilan *landmark* wajah dan menangani input dari kamera [4].

5.2 MediaPipe

Library ini merupakan salah satu *framework* dari *Google* yang berguna untuk pemrosesan media berbasis AI , terutama untuk deteksi *landmark* wajah dan pelacakan gerakan. Pada project ini di file facedetect.py modul mediapipe.solutions.face_mesh berguna untuk medeteksi titik-titik diwajah (468 titik), dan mediapipe.solutions.drawing_utils berguna untuk menggambar koneksi antar titik di wajah [3].

5.3 Numpy

Library Numpy merupakan pustaka Python untuk komputasi numerik dan manipulasi array multidimensi.Pada file expression.py Numpy digunakan untuk melakukan perhitungan dasar seperti pengolahan posisi pixel [5, 6].

5.4 tkinter

Library Tkinter ini merupakan library antarmuka GUI bawahaan python yang digunakan pada file **gui.py** dan **expression.py**. Library ini digunakan untuk menampilkan jendela seleksi kamera, tombol untuk memulai aplikasi, dan menyembunyikan jendela utama agar tidak mengganggu tampilan saat terjadi pop-up [7].

5.5 ttk (Themed Tkinter Widgets)

Bagian dari tkinter, modul ttk memberikan akses ke widget bertema modern (seperti Combobox, Label, Button, dan Frame) dengan tampilan lebih konsisten di berbagai platform. Pada **gui.py,** ttk digunakan untuk menyusun antarmuka pengguna agar lebih menarik, termasuk penggunaan custom style seperti font Poppins dan pengaturan warna tombol.

5.6 from ... import ... (Import Antar Modul Sendiri)

Program ini menggunakan modularisasi kode, dengan mengimpor modul seperti facedetect, expression, gui, utils, dan game. Contohnya, **expression.py** mengimpor FaceDetector dari facedetect, dan **gui.py** mengimpor fungsi list_available_cameras dari utils. Ini merupakan salah satu aspek clean code yaitu dengan membagi tanggung jawab fungsional ke dalam file terpisah, meningkatkan keterbacaan, dan memudahkan pemeliharaan.

6 Hasil Analisis

Pada bagian ini, dianalisis program utama yang bertanggung jawab untuk inisialisasi dan menjalankan keseluruhan alur aplikasi "Swipe That Face!".

6.1 Program Utama (`main.py`)

Bagian program utama, yang terdapat dalam file `main.py`, bertugas sebagai titik masuk aplikasi. Program ini mengorkestrasi pemanggilan modul antarmuka awal dan peluncuran jendela permainan utama. Berikut adalah implementasi dari bagian inti program tersebut:

```python
from PyQt6.QtWidgets import QApplication
from gui import launch_camera_selector
from qtgame import GameWindow
import sys

def main():
    app = QApplication(sys.argv)
    cam_index = launch_camera_selector()
    if cam_index is not None:
        window = GameWindow(cam_index)
        window.show()
        sys.exit(app.exec())

if __name__ == "__main__":
    main()
```

Kode 1: Struktur Utama main.py

6.1.1 Deskripsi Kode

Kode ini melakukan langkah-langkah berikut:

1. **Mengontrol Ekseskusi**:
   - Kode diawali dengan pernyataan `if __name__ == "__main__:"`. Pernyataan ini memastikan bahwa fungsi `main()` hanya dieksekusi ketika file `main.py` dijalankan secara langsung oleh interpreter Python, dan bukan ketika diimpor sebagai modul oleh skrip lain. Ini adalah praktik standar untuk mengontrol eksekusi kode utama.
2. **Inisialisasi PyQt6**:

- Di dalam fungsi `main()`, baris `app = QApplication(sys.argv)` membuat instance dari kelas `QApplication`. Objek ini diperlukan untuk setiap aplikasi GUI yang menggunakan pustaka PyQt6, karena bertugas mengelola *event loop* dan sumber daya global aplikasi.

3. **Camera Launcher**:

   - Fungsi `launch_camera_selector()` yang diimpor dari modul `gui.py` kemudian dipanggil. Fungsi ini bertanggung jawab untuk menampilkan antarmuka pengguna (berbasis Tkinter) yang memungkinkan pengguna memilih sumber kamera yang akan digunakan. Indeks kamera yang dipilih dikembalikan dan disimpan dalam variabel `cam_index`.

4. **Validasi dan Memulai Permainan**:

   - Dilakukan pengecekan `if cam_index is not None:`. Jika pengguna telah memilih kamera yang valid (artinya `cam_index` bukan `None`), program akan melanjutkan untuk membuat instance dari kelas `GameWindow` (`window = GameWindow(cam_index)`). Kelas `GameWindow`, yang diimpor dari modul `qtgame.py`, merepresentasikan jendela utama permainan dan menerima indeks kamera sebagai argumen.
   - Metode `window.show()` kemudian dipanggil untuk menampilkan jendela permainan kepada pengguna.

5. ***Loop* Permainan**:

   - Terakhir, `sys.exit(app.exec())` dieksekusi. Perintah `app.exec()` memulai *event loop* utama dari PyQt6, yang membuat aplikasi tetap berjalan, merespons interaksi pengguna, dan menangani berbagai *event* hingga jendela utama ditutup. Fungsi `sys.exit()` memastikan program keluar dengan bersih. Jika tidak ada kamera yang dipilih, blok ini akan dilewati, dan program akan berakhir tanpa meluncurkan jendela permainan.

Dengan langkah-langkah tersebut, `main.py` berhasil mengintegrasikan modul antarmuka pemilihan kamera (`gui.py`) dengan modul jendela permainan utama (`qtgame.py`) untuk menyajikan fungsionalitas aplikasi "Swipe That Face!" kepada pengguna.

7 Implementasi Program

7.1 Langkah 1: Menyalin Link Repository

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

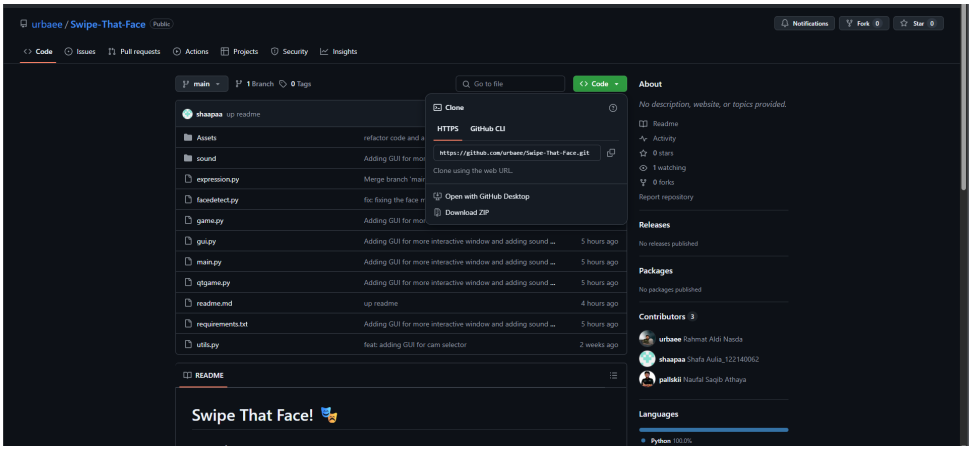
1. Buka halaman repository GitHub melalui tautan berikut:

   ```
   \url{https://github.com/urbaee/Swipe-That-Face}.
   ```

2. Klik tombol **Code** berwarna hijau di bagian atas repository.

3. Pilih opsi **HTTPS** dan salin URL yang ditampilkan, yaitu:

   ```
   https://github.com/urbaee/Swipe-That-Face.git
   ```



Gambar 1: Salin URL Repository

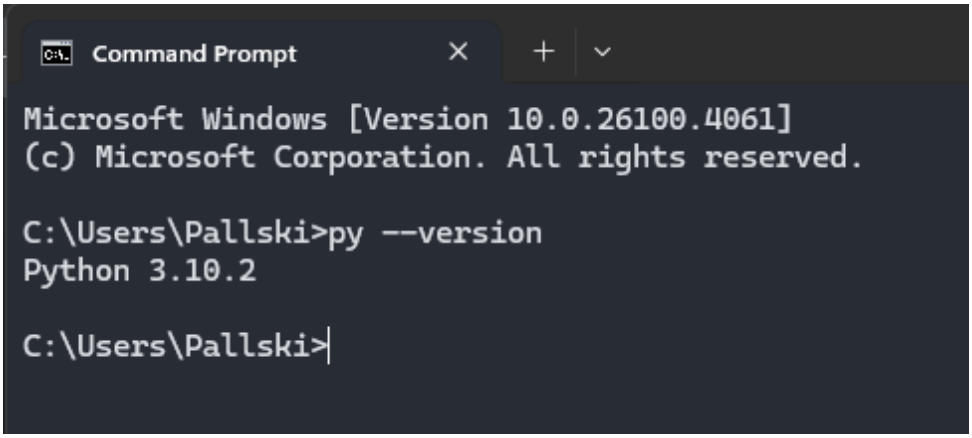
7.2 Langkah 2: Memeriksa Versi Python

Untuk memastikan program dapat dijalankan dengan benar, diperlukan Python versi 3.10 hingga 3.12. Berikut adalah langkah-langkah untuk mengecek versi Python:

1. Buka terminal atau command prompt.
2. Ketik perintah berikut:

```
py --version
```

3. Pastikan versi Python yang ditampilkan berada dalam rentang 3.10 hingga 3.12. Jika versi Python tidak sesuai, instal Python versi yang sesuai terlebih dahulu.



Gambar 2: Version Python Check

7.3 Langkah 3: Melakukan Clone Repository

Setelah memastikan versi Python, langkah berikutnya adalah melakukan clone repository. Langkah-langkahnya adalah sebagai berikut:

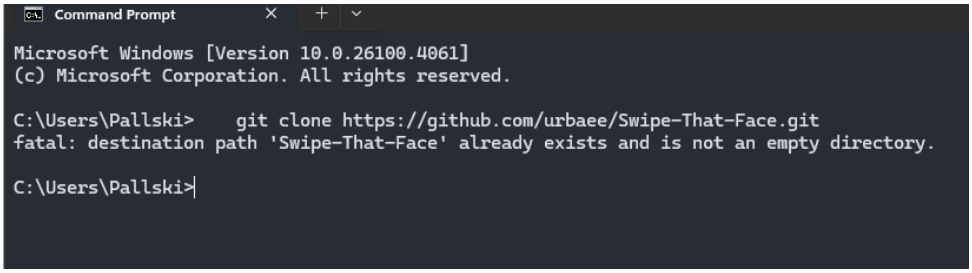
1. Buka terminal atau command prompt.
2. Pindah ke direktori tujuan untuk menyimpan repository dengan perintah:

```
cd /path/to/directory
```

3. Clone repository dengan perintah:

```
git clone https://github.com/urbaee/Swipe-That-Face.git
```

4. Setelah proses selesai, sebuah direktori baru dengan nama repository akan muncul di dalam direktori tujuan.

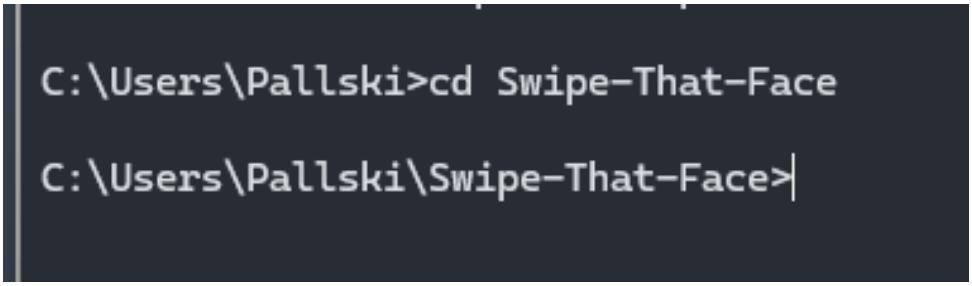


Gambar 3: Apply Clone Repository

7.4 Langkah 4: Pindah ke Direktori Repository

Untuk mengakses file repository yang telah di-clone, pindah ke direktori repository dengan perintah:

```
cd Swipe-That-Face
```



Gambar 4: Pindah ke Path Clone

7.5 Langkah 5: Menginstal Dependensi

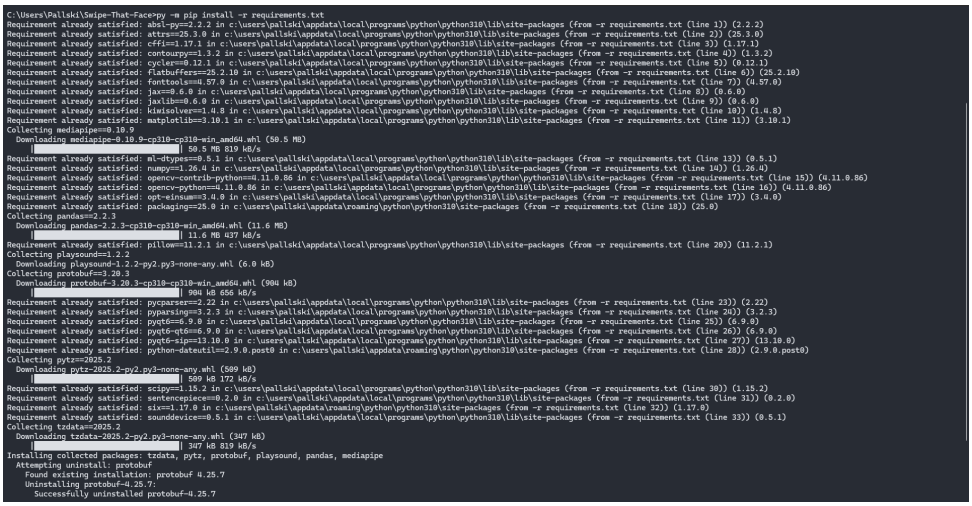
File `requirements.txt` berisi daftar library yang diperlukan untuk menjalankan program. Instal semua dependensi dengan perintah berikut:

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

Disini kami menyarankan anda agar dapat menggunakan virtual environtment seperti uv atau conda untuk menghindari tabrakan paket.

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

Pastikan semua library terinstal tanpa error sebelum melanjutkan.



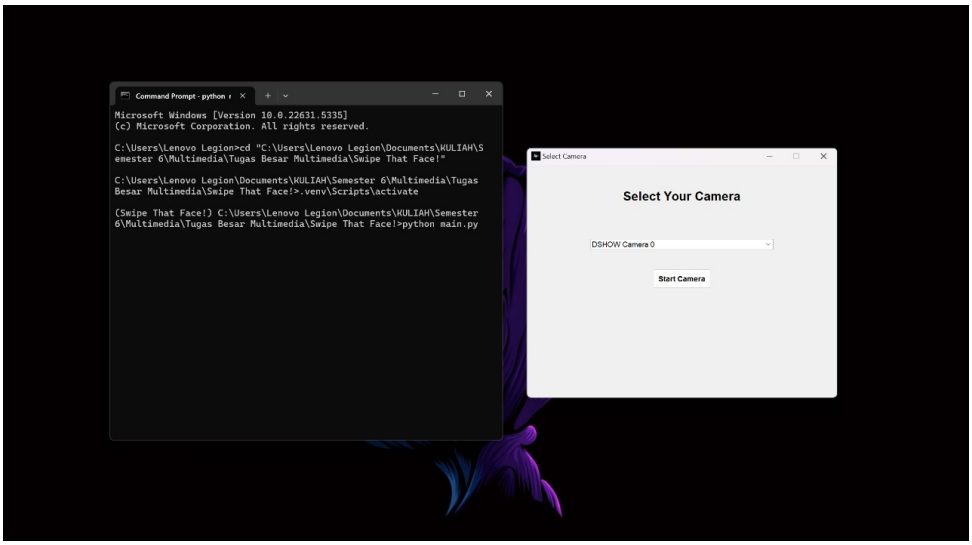
Gambar 5: Install Requirements

7.6 Langkah 6: Menjalankan Program

Setelah dependensi terinstal, jalankan program dengan perintah berikut:

```
python main.py
```

Program akan mulai berjalan, dan antarmuka permainan akan ditampilkan.



Gambar 6: Menjalankan Program

7.7 Langkah 5: Antarmuka Aplikasi dan Penggunaan

Setelah aplikasi "Swipe That Face!" berhasil dijalankan dan kamera dipilih, pengguna akan berinteraksi dengan antarmuka utama permainan. Antarmuka ini menyediakan beberapa tombol kontrol utama:

1. **Start Game**: Digunakan untuk memulai sesi permainan. Setelah ditekan, akan ada hitung mundur singkat sebelum tantangan ekspresi pertama ditampilkan.

2. **Toggle Landmarks**: Memungkinkan pengguna untuk menampilkan atau menyembunyikan titik-titik *landmark* wajah yang dideteksi oleh sistem pada tampilan video. Fitur ini berguna untuk melihat bagaimana sistem mendeteksi wajah.

3. **Reset Game**: Tombol ini akan aktif setelah satu putaran permainan selesai. Jika ditekan, permainan akan kembali ke kondisi awal, dan pengguna bisa memulai tantangan dari awal lagi.

4. **Quit**: Berfungsi untuk menutup dan keluar dari aplikasi "Swipe That Face!".

Selain tombol-tombol tersebut, area utama layar akan menampilkan video dari kamera pengguna, emoji yang harus ditiru, serta pesan status permainan (seperti countdown, "Kamu berhasil!", atau "Game Completed!").

7.7.1 Tampilan Antarmuka

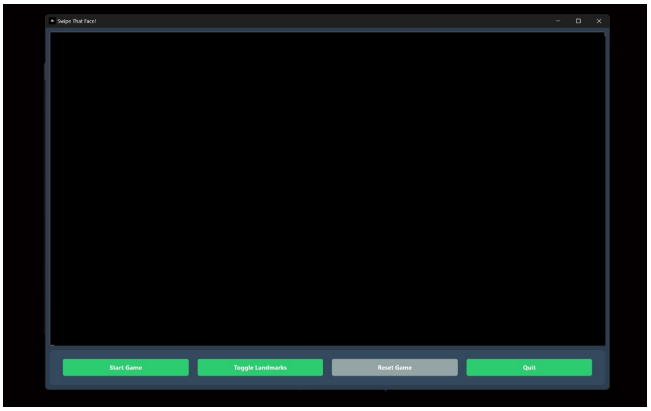
Berikut adalah beberapa tangkapan layar dari antarmuka program:

- **Start Game:**

Gambar 7: Start Game

Gambar 7 merupakan **Start Game** dari permainan Swipe That Face yang dikendalikan menggunakan MediaPipe.Tombol ini untuk memulai permainan dengan menekan tombol `Start Game`

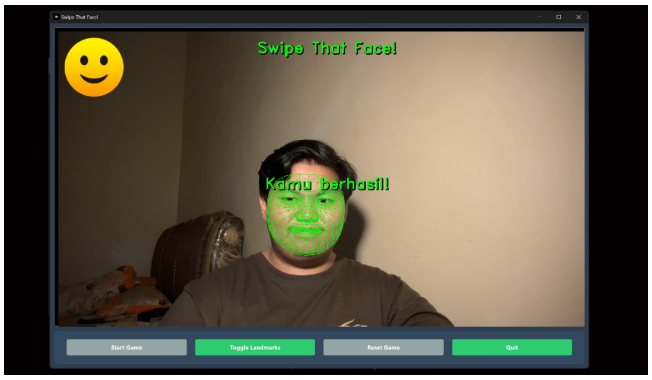
- **Preview:**



Gambar 8: Preview

Gambar 8 merupakan **Preview** dari permainan Swipe That Face tanpa adanya input kamera.

- **Toggle Landmark:**



Gambar 9: Toggle Landmark

Gambar 9 ini merupakan fitur **Toggle Landmark** yang gunanya untuk menampilkan atau menyembunyikan landmark di wajah user saat bermain Swipe That Face. Kalau diaktifkan, pengguna bisa lihat bagaimana sistem membaca wajah mereka dengan landmark. Kalau mau tampilan video lebih bersih, tinggal tekan lagi tombolnya untuk menghilangkan landmarknya.

- **Menang Permainan**

Gambar 10: Menang Permainan

Gambar 10 merupakan UI ketika player dapat memenangkan permainan dengan notifikasi **Kamu Berhasil!**. Setelah itu player dapat melakukan **Reset Game** agar dapat memainkan kembali permainan Swipe That Face lagi.

8 Referensi dan Daftar Pustaka

References

[1] Python Software Foundation, "Python programming language," https://www.python.org, 2025.

[2] R. Erlangga, R. Silvana, and R. Regasari, "Penerapan metode pemrograman berorientasi objek pada bahasa pemrograman python dalam merancang aplikasi praktik mandiri bidan rozi silvana," *Jurnal Teknologi dan Sistem Informasi (JTSI)*, 2024, available at: https://www.researchgate.net/profile/Reuben-Erlangga/publication/388330210_PENERAPAN_METODE_PEMROGRAMAN_BERORIENTASI_OBJEK_PADA_

BAHASA_PEMROGRAMAN_PYTHON_DALAM_MERANCANG_APLIKASI_PRAKTIK_MANDIRI_BIDAN_ROZI_SILVANA/links/679330538311ce680c324caa/PENERAPAN-METODE-PEMROGRAMAN-BERORIENTASI-OBJEK-PADA-BAHASA-PEMROGRAMAN_PYTHON_DALAM_MERANCANG_APLIKASI_PRAKTIK_MANDIRI_BIDAN_ROZI_SILVANA.pdf. Accessed: May 30, 2025.

[3] Google, "Mediapipe," https://mediapipe.dev, 2025.

[4] OpenCV Team, "Opencv library," https://opencv.org, 2025.

[5] NumPy Developers, "Numpy - the fundamental package for scientific computing with python," https://numpy.org/, 2025, accessed: May 24, 2025.

[6] S. Anam, *Pengantar Algoritma dan Pemrograman dengan Python*. UB Press, 2023.

[7] Python Software Foundation, "Tkinter — python interface to tcl/tk," https://docs.python.org/3/library/tkinter.html, 2025.

Link Tambahan

Proses penyusunan beberapa bagian laporan ini memanfaatkan sumber daya digital seperti berikut:

- **Bantuan Penyusunan Draf Laporan (ChatGPT):**
  - Diskusi mengenai draf untuk bagian implementasi program:
    `https://chatgpt.com/share/6839ccee-3b40-8013-9bd6-2b4e3d76bed8`
    `https://chatgpt.com/share/6839cd47-b650-8013-8506-2f2406811e3f`
- **Lampiran percakapan LLM (Google Drive):**
  `https://drive.google.com/drive/folders/1MeXkXNZ-6Hq6_iyg9zlTWJah6AAdnGph?usp=sharing`
- **Link repository github:**
  `https://github.com/urbaee/Swipe-That-Face`
- **Link youtube demo:**
  `https://youtu.be/mJ7a0TqJscs`