



Nama: **Rahmat Aldi Nasda, Shafa Aulia, Naufal Saqib Athaya**
Mata Kuliah: **Sistem/Teknologi Multimedia (IF4021)**

Tugas: **Final Project**
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 mencocokkan 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.

- **PyQt6**

PyQt6 adalah serangkaian binding *Python* untuk framework aplikasi *Qt* versi 6 yang dikembangkan oleh The Qt Company. Binding ini diimplementasikan sebagai kumpulan modul *Python* yang mencakup lebih dari 1.000 kelas. PyQt6 memungkinkan pengembang untuk membuat aplikasi GUI lintas platform menggunakan bahasa *Python*. Aplikasi yang dibangun dengan PyQt6 dapat dijalankan di berbagai sistem operasi seperti *Windows*, *macOS*, dan *Linux* [8]. Dalam proyek ini, PyQt6 digunakan untuk:

- Membangun antarmuka pengguna yang modern dan responsif.
- Mengelola berbagai widget seperti tombol, label, dan input teks.
- Mengatur layout dan navigasi antar jendela aplikasi.

- **Webcam**

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

- **FaceMesh Detection Model**

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**, **facedetector.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.

```

1 import cv2
2 import numpy as np
3 import tkinter as tk
4 from facedetector import FaceDetector
5
6 tk.Tk().withdraw() # Hide main tkinter window for popup use
7
8 def is_smiling(landmarks, image_width, image_height):
9     """Cek apakah user tersenyum berdasarkan FaceMesh landmark."""
10
11     # Ambil titik-titik yang relevan
12     def get_coord(idx):
13         pt = landmarks.landmark[idx]
14         return int(pt.x * image_width), int(pt.y * image_height)
15
16     # Titik penting
17     left_corner = get_coord(61)
18     right_corner = get_coord(291)
19     mid_upper_lip = get_coord(13)
20     bottom_lip = get_coord(14)
21
22     # Mulut harus tertutup (opsional tapi penting)

```

```

23     mouth_open = abs(bottom_lip[1] - mid_upper_lip[1])
24     is_mouth_closed = mouth_open < 5
25
26     # Senyum: sudut bibir lebih tinggi dari titik tengah
27     left_is_higher = left_corner[1] < mid_upper_lip[1] - 3
28     right_is_higher = right_corner[1] < mid_upper_lip[1] - 3
29     is_corner_up = left_is_higher and right_is_higher
30
31     return is_mouth_closed and is_corner_up
32
33 def is_big_smiling(landmarks, image_width, image_height):
34     """Detect big smile by measuring vertical lip distance ratio"""
35     def get_coord(idx):
36         pt = landmarks.landmark[idx]
37         return int(pt.x * image_width), int(pt.y * image_height)
38
39     left_mouth, right_mouth = get_coord(61), get_coord(291)
40     top_lip, bottom_lip = get_coord(13), get_coord(14)
41
42     mouth_width = abs(right_mouth[0] - left_mouth[0])
43     mouth_height = abs(bottom_lip[1] - top_lip[1])
44
45     smile_ratio = mouth_width / (mouth_height + 1e-5)
46
47     # Big smile = mulut terbuka cukup tinggi, tapi tetap lebar
48     return smile_ratio > 1.8 and mouth_height > 20
49
50 def is_blinking(landmarks, image_width, image_height):
51     """Check if user is blinking (either left or right eye)"""
52     def get_coord(idx):
53         pt = landmarks.landmark[idx]
54         return int(pt.x * image_width), int(pt.y * image_height)
55
56     # Get eye landmarks
57     left_eye_top = get_coord(159)
58     left_eye_bottom = get_coord(145)
59     right_eye_top = get_coord(386)
60     right_eye_bottom = get_coord(374)
61
62     # Calculate eye aspect ratios
63     left_eye_height = abs(left_eye_top[1] - left_eye_bottom[1])
64     right_eye_height = abs(right_eye_top[1] - right_eye_bottom[1])
65
66     # Check if either eye is closed (small height indicates closed eye)
67     threshold = 5
68     return left_eye_height < threshold or right_eye_height < threshold
69
70 def is_eyes_closed(landmarks, image_width, image_height):
71     """Check if both eyes are closed"""
72     def get_coord(idx):
73         pt = landmarks.landmark[idx]
74         return int(pt.x * image_width), int(pt.y * image_height)
75
76     # Get eye landmarks for both eyes
77     left_eye_top = get_coord(159)
78     left_eye_bottom = get_coord(145)
79     right_eye_top = get_coord(386)
80     right_eye_bottom = get_coord(374)
81
82     # Calculate eye heights
83     left_eye_height = abs(left_eye_top[1] - left_eye_bottom[1])
84     right_eye_height = abs(right_eye_top[1] - right_eye_bottom[1])

```

```

85
86     # Mata (booth) harus nutup
87     threshold = 5
88     return left_eye_height < threshold and right_eye_height < threshold
89
90 def overlay_img(background, overlay, pos):
91     """Menggabungkan overlay (dengan transparansi) ke frame utama."""
92     x, y = pos
93     h, w = overlay.shape[:2]
94
95     # Cek apakah ada alpha channel
96     if overlay.shape[2] == 4:
97         alpha = overlay[:, :, 3] / 255.0
98         for c in range(3):
99             background[y:y+h, x:x+w, c] = (
100                 alpha * overlay[:, :, c] +
101                 (1 - alpha) * background[y:y+h, x:x+w, c]
102             )
103     else:
104         background[y:y+h, x:x+w] = overlay
105
106 def show_expression_game(cam_index: int, emoji_path: str):
107     from PyQt6.QtWidgets import QApplication
108     from qtgame import GameWindow
109     import sys
110
111     app = QApplication(sys.argv)
112     window = GameWindow(cam_index)
113     window.show()
114     sys.exit(app.exec())

```

4.1.1 Fungsi `is_smiling()`

- 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_smiling()`

- 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 melihat 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 sebelumnya 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 *landmark* 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.

```

1 import mediapipe as mp
2 import cv2
3
4 class FaceDetector:
5     def __init__(self, max_faces=1):
6         self.mp_face_mesh = mp.solutions.face_mesh
7         self.mp_drawing = mp.solutions.drawing_utils
8         self.face_mesh = self.mp_face_mesh.FaceMesh(
9             static_image_mode=False,
10            max_num_faces=max_faces,
11            refine_landmarks=True,
12            min_detection_confidence=0.5,
13            min_tracking_confidence=0.5
14        )
15
16    def get_landmarks(self, frame):
17        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
18        results = self.face_mesh.process(rgb_frame)
19        return results.multi_face_landmarks
20
21    def draw_landmarks(self, frame, landmarks):
22        # Draw title text
23        ih, iw, _ = frame.shape
24        title = "Swipe That Face!"
25        font = cv2.FONT_HERSHEY_DUPLEX # Closest to Poppins available in OpenCV
26        font_scale = 1.2
27        thickness = 2
28        text_size = cv2.getTextSize(title, font, font_scale, thickness)[0]
29        text_x = (iw - text_size[0]) // 2 # Center horizontally
30        text_y = 50 # Position from top
31
32        # Draw text shadow/outline for better visibility
33        cv2.putText(frame, title, (text_x+2, text_y+2), font, font_scale, (0, 0, 0), thickness+1)
34        cv2.putText(frame, title, (text_x, text_y), font, font_scale, (0, 255, 0), thickness)
35
36        # Draw landmarks
37        for face_landmarks in landmarks:
38            self.mp_drawing.draw_landmarks(
39                image=frame,
40                landmark_list=face_landmarks,
41                connections=self.mp_face_mesh.FACEMESH_TESSELATION,
42                landmark_drawing_spec=None,
43                connection_drawing_spec=self.mp_drawing.DrawingSpec(
44                    color=(0, 255, 0),
45                    thickness=1,
46                    circle_radius=1
47                )
48            )
49        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 minimum 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 ditampilkan 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.

```

1 import cv2
2 import numpy as np
3 from facedetect import FaceDetector
4 from expression import is_smiling, is_big_smiling, is_blinking, is_eyes_closed
5 from playsound import playsound
6 import threading
7 import sys
8
9 class ExpressionGame:
10     def __init__(self, cam_index):
11         self.cap = cv2.VideoCapture(cam_index)
12         # Set camera properties to 1280x720
13         self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
14         self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
15         self.detector = FaceDetector()
16         self.window_size = (1280, 720) # Changed from (640, 640)
17         self.setup_window = lambda: None # Disable CV2 window creation
18         self.setup_game_states()
19         self.load_expressions()

```

```

20     self.success_sound = "sound/berhasil.mp3"
21     self.show_landmarks = True # Add this line to track landmark visibility
22     self.audio_threads = [] # Add this line to track audio threads
23
24     def setup_game_states(self):
25         self.MENU = 0
26         self.COUNTDOWN = 1
27         self.PLAYING = 2
28         self.game_state = self.MENU
29         self.countdown_start = None
30         self.countdown_duration = 1.0
31         self.win_start_time = None
32         self.WIN_DURATION = 3
33         self.game_completed = False
34         self.current_expression = 0
35
36     def load_expressions(self):
37         self.expressions = [
38             {"emoji": "Assets/senyum.png", "detector": is_smiling},
39             {"emoji": "Assets/senyumlebar.png", "detector": is_big_smiling},
40             {"emoji": "Assets/kedip.png", "detector": is_blinking},
41             {"emoji": "Assets/merem.png", "detector": is_eyes_closed},
42         ]
43         self.current_emoji = cv2.imread(self.expressions[0]["emoji"], cv2.IMREAD_UNCHANGED)
44         if self.current_emoji is None:
45             raise FileNotFoundError("Emoji image not found!")
46         self.current_emoji = cv2.resize(self.current_emoji, (150, 150))
47
48     def reset_game(self):
49         self.current_expression = 0
50         self.win_start_time = None
51         self.game_completed = False
52         self.current_emoji = cv2.imread(self.expressions[0]["emoji"], cv2.IMREAD_UNCHANGED)
53         self.current_emoji = cv2.resize(self.current_emoji, (150, 150))
54
55     def draw_ui(self, frame, text, position, font_scale=1.2, color=(0, 255, 0)):
56         h, w = frame.shape[:2]
57         font = cv2.FONT_HERSHEY_DUPLEX
58         thickness = 2
59         text_size = cv2.getTextSize(text, font, font_scale, thickness)[0]
60
61         if position == 'center':
62             text_x = (w - text_size[0]) // 2
63             text_y = (h + text_size[1]) // 2
64         elif position == 'top':
65             text_x = (w - text_size[0]) // 2
66             text_y = 50
67         elif position == 'bottom':
68             text_x = (w - text_size[0]) // 2
69             text_y = h - 50
70
71         cv2.putText(frame, text, (text_x+2, text_y+2), font, font_scale, (0, 0, 0), thickness+1)
72         cv2.putText(frame, text, (text_x, text_y), font, font_scale, color, thickness)
73
74     def show_main_menu(self, frame):
75         self.draw_ui(frame, "Swipe That Face!", 'top', 1.5, (0, 255, 0))
76         # Removed "Press SPACE to PLAY" text
77
78     def show_countdown(self, frame, number):
79         self.draw_ui(frame, str(number), 'center', 4.0, (0, 255, 255))
80
81     def overlay_emoji(self, frame, emoji, pos):

```



```

82     x, y = pos
83     h, w = emoji.shape[:2]
84     if emoji.shape[2] == 4:
85         alpha = emoji[:, :, 3] / 255.0
86         for c in range(3):
87             frame[y:y+h, x:x+w, c] = (
88                 alpha * emoji[:, :, c] +
89                 (1 - alpha) * frame[y:y+h, x:x+w, c]
90             )
91     else:
92         frame[y:y+h, x:x+w] = emoji
93
94 def cleanup(self):
95     """Cleanup resources before exit"""
96     if self.cap is not None:
97         self.cap.release()
98     cv2.destroyAllWindows()
99
100     # Wait for any audio threads to complete
101     for thread in self.audio_threads:
102         if thread.is_alive():
103             thread.join(timeout=1.0)
104
105 def play_success_sound(self):
106     """Play sound in thread and track it"""
107     thread = threading.Thread(
108         target=playsound,
109         args=(self.success_sound,),
110         daemon=True
111     )
112     self.audio_threads.append(thread)
113     thread.start()
114
115 def get_current_frame(self):
116     """Get processed frame for current game state"""
117     success, frame = self.cap.read()
118     if not success:
119         return None
120
121     # Process frame
122     frame = self.process_frame(frame)
123
124     # Handle different game states
125     if self.game_state == self.MENU:
126         self.show_main_menu(frame)
127     elif self.game_state == self.COUNTDOWN:
128         self.handle_countdown(frame)
129     elif self.game_state == self.PLAYING:
130         self.handle_gameplay(frame)
131
132     return frame
133
134 def play_success_sound(self):
135     # Play in separate thread to avoid blocking
136     threading.Thread(target=playsound, args=(self.success_sound,), daemon=True).start()
137
138 def process_frame(self, frame):
139     """Process the frame for gameplay"""
140     h, w = frame.shape[:2]
141     target_w, target_h = self.window_size
142
143     # Calculate scaling factor

```

```

144     scale = min(target_w/w, target_h/h)
145     new_w = int(w * scale)
146     new_h = int(h * scale)
147
148     # Resize frame preserving aspect ratio
149     frame = cv2.resize(frame, (new_w, new_h))
150
151     # Create black canvas of target size
152     canvas = np.zeros((target_h, target_w, 3), dtype=np.uint8)
153
154     # Center the frame on canvas
155     y_offset = (target_h - new_h) // 2
156     x_offset = (target_w - new_w) // 2
157     canvas[y_offset:y_offset+new_h, x_offset:x_offset+new_w] = frame
158
159     return canvas
160
161 def start_game(self):
162     """Start game from menu state"""
163     if self.game_state == self.MENU:
164         self.game_state = self.COUNTDOWN
165         self.countdown_start = cv2.getTickCount()
166
167 def handle_gameplay(self, frame):
168     h, w = frame.shape[:2]
169     landmarks_list = self.detector.get_landmarks(frame)
170
171     self.draw_ui(frame, "Swipe That Face!", 'top')
172
173     if landmarks_list:
174         if self.show_landmarks:
175             self.detector.draw_landmarks(frame, landmarks_list)
176
177         if not self.game_completed:
178             expression_detected = self.expressions[self.current_expression]["detector"](
179                 landmarks_list[0], w, h)
180             if expression_detected and self.win_start_time is None:
181                 self.win_start_time = cv2.getTickCount()
182
183     self.handle_win_state(frame)
184
185     if not self.game_completed:
186         self.overlay_emoji(frame, self.current_emoji, (10, 10))
187     else:
188         # Changed message to be simpler
189         self.draw_ui(frame, "Game Completed!", position='center', font_scale=1.0, color=(255,
190 255, 255))
191
192 def handle_countdown(self, frame):
193     """Handle countdown state"""
194     current_time = cv2.getTickCount()
195     elapsed = (current_time - self.countdown_start) / cv2.getTickFrequency()
196     countdown_number = 3 - int(elapsed)
197
198     if countdown_number > 0:
199         self.show_countdown(frame, countdown_number)
200     else:
201         self.game_state = self.PLAYING
202         self.reset_game()
203
204 def handle_win_state(self, frame):
205     """Handle win state logic"""

```

```

205     if self.win_start_time is not None and not self.game_completed:
206         current_time = cv2.getTickCount()
207         elapsed = (current_time - self.win_start_time) / cv2.getTickFrequency()
208
209         if elapsed <= self.WIN_DURATION:
210             self.draw_ui(frame, "Kamu berhasil!", 'center')
211             # Play sound when first entering win state
212             if elapsed < 0.1: # Only play once at the start of win state
213                 self.play_success_sound()
214         else:
215             if self.current_expression < len(self.expressions) - 1:
216                 self.current_expression += 1
217                 self.current_emoji = cv2.imread(
218                     self.expressions[self.current_expression]["emoji"],
219                     cv2.IMREAD_UNCHANGED
220                 )
221                 self.current_emoji = cv2.resize(self.current_emoji, (150, 150))
222             else:
223                 self.game_completed = True
224                 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 objek 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 variabel 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 modul 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)`

- Mengubah 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**.

```

1 import tkinter as tk
2 from tkinter import ttk, PhotoImage
3 import cv2
4 from facedetect import FaceDetector
5 from utils import list_available_cameras
6 from expression import show_expression_game
7 from PyQt6.QtGui import QIcon
8
9 def run_landmark_view(cam_index):
10     cap = cv2.VideoCapture(cam_index)
11     detector = FaceDetector()
12
13     while cap.isOpened():
14         success, frame = cap.read()
15         if not success:
16             print("Gagal ambil frame.")
17             break
18
19         landmarks = detector.get_landmarks(frame)
20         if landmarks:
21             detector.draw_landmarks(frame, landmarks)
22
23         cv2.imshow(f'Face Landmark Viewer - Camera {cam_index}', frame)
24
25         if cv2.waitKey(1) & 0xFF == ord('q'):
26             break
27
28     cap.release()
29     cv2.destroyAllWindows()
30
31 def get_available_cameras():
32     """Deteksi kamera yang benar-benar tersedia"""
33     available_cameras = []
34     max_test = 10 # Test sampai index 10
35
36     for i in range(max_test):
37         cap = cv2.VideoCapture(i, cv2.CAP_DSHOW) # Gunakan DirectShow
38         if cap.isOpened():
39             # Ambil informasi kamera
40             ret, frame = cap.read()
41             if ret:
42                 # Dapatkan nama device
43                 name = f"Camera {i}"
44                 try:
45                     # Coba dapatkan nama backend
46                     backend = cap.getBackendName()
47                     if backend:
48                         name = f"{backend} Camera {i}"
49                 except:
50                     pass
51                 available_cameras.append((i, name))
52         cap.release()
53     return available_cameras
54
55 class CameraSelector(tk.Toplevel):
56     def __init__(self, parent=None):

```

```

57     super().__init__(parent)
58     self.selected_camera = None
59     self.camera_indices = []
60     # Change icon setting method
61     self.icon = PhotoImage(file='logo/icon.png')
62     self.tk.call('wm', 'iconphoto', self._w, self.icon)
63     self.initUI()
64     self.center_window()
65
66 def center_window(self):
67     # Get screen width and height
68     screen_width = self.winfo_screenwidth()
69     screen_height = self.winfo_screenheight()
70
71     # Calculate window position
72     x = (screen_width - 800) // 2 # 800 is window width
73     y = (screen_height - 600) // 2 # 600 is window height
74
75     # Set window position
76     self.geometry(f'800x600+{x}+{y}')
77
78 def initUI(self):
79     self.title('Select Camera')
80     self.geometry('800x600') # Ukuran window lebih besar
81     self.resizable(False, False)
82
83     # Style untuk judul
84     title_label = tk.Label(
85         self,
86         text="Select Your Camera",
87         font=('Arial', 24, 'bold'), # Font lebih besar
88         pady=20
89     )
90     title_label.pack(pady=40) # Spacing lebih besar
91
92     # Detect available cameras
93     cameras = get_available_cameras()
94
95     if not cameras:
96         label = tk.Label(
97             self,
98             text="No cameras found!",
99             font=('Arial', 16), # Font lebih besar
100             fg='red'
101         )
102         label.pack(pady=30)
103     else:
104         # Store camera indices and names
105         self.camera_indices = [idx for idx, name in cameras]
106         camera_names = [name for idx, name in cameras]
107
108         # Style combobox
109         style = ttk.Style()
110         style.configure('Custom.TCombobox', font=('Arial', 14)) # Font lebih besar
111
112         self.combo = ttk.Combobox(
113             self,
114             values=camera_names,
115             state='readonly',
116             width=40, # Width lebih besar
117             font=('Arial', 14), # Font lebih besar
118             style='Custom.TCombobox'

```

```

119         )
120         self.combo.pack(pady=(30, 20)) # Spacing lebih besar
121         self.combo.current(0)
122
123         # Style button
124         select_btn = ttk.Button(
125             self,
126             text='Start Camera', # Text lebih deskriptif
127             command=self.on_select,
128             style='Custom.TButton'
129         )
130
131         # Custom style untuk button
132         style.configure(
133             'Custom.TButton',
134             font=('Arial', 14, 'bold'), # Font lebih besar
135             padding=10
136         )
137
138         select_btn.pack(pady=30) # Spacing lebih besar
139
140     def on_select(self):
141         current_idx = self.combo.current() # Use current() instead of currentIndex()
142         self.selected_camera = self.camera_indices[current_idx] # Get the actual camera index
143         self.destroy()
144
145     def launch_camera_selector():
146         dialog = CameraSelector()
147         dialog.grab_set() # Make the dialog modal
148         dialog.wait_window() # Wait for the dialog to close
149         return dialog.selected_camera if dialog.selected_camera is not None else -1
150
151     def main():
152         cam_index = launch_camera_selector()
153         if cam_index != -1:
154             run_landmark_view(cam_index)
155
156     if __name__ == "__main__":
157         root = tk.Tk()
158         root.withdraw() # Hide the root window
159         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 proyek yang kami kembangkan "**Swipe That Face!**" yang dibangun menggunakan **PyQt6**.

```

1 from PyQt6.QtWidgets import (QMainWindow, QWidget, QVBoxLayout,
2                               QHBoxLayout, QPushButton, QLabel, QFrame)
3 from PyQt6.QtCore import Qt, QTimer
4 from PyQt6.QtGui import QImage, QPixmap, QIcon
5 import cv2
6 import threading
7 from playsound import playsound
8 import numpy as np
9 from game import ExpressionGame
10
11 class GameWindow(QMainWindow):
12     def __init__(self, cam_index):
13         super().__init__()
14         self.game = ExpressionGame(cam_index)
15         self.is_closing = False
16         # Start background music in a separate thread
17         self.bg_music_thread = threading.Thread(

```



```

18         target=self.play_background_music,
19         daemon=True
20     )
21     self.bg_music_thread.start()
22     self.setWindowIcon(QIcon('logo/icon.png'))
23     self.initUI()
24
25     def play_background_music(self):
26         while not self.is_closing:
27             playsound("sound/backsound.mp3")
28
29     def closeEvent(self, event):
30         self.is_closing = True
31         if hasattr(self, 'timer'):
32             self.timer.stop()
33         self.game.cleanup()
34         event.accept()
35
36     def initUI(self):
37         # Main widget and layout
38         central_widget = QWidget()
39         self.setCentralWidget(central_widget)
40         main_layout = QVBoxLayout(central_widget)
41         main_layout.setSpacing(10) # Add space between containers
42
43         # Video container
44         video_container = QFrame()
45         video_container.setFrameStyle(QFrame.Shape.Box)
46         video_container.setStyleSheet("QFrame { background-color: black; }")
47         video_layout = QVBoxLayout(video_container)
48
49         # Video feed label
50         self.video_label = QLabel()
51         self.video_label.setMinimumSize(1280, 720)
52         self.video_label.setAlignment(Qt.AlignmentFlag.AlignCenter)
53         video_layout.addWidget(self.video_label)
54
55         # Add video container to main layout
56         main_layout.addWidget(video_container)
57
58         # Button container
59         button_container = QFrame()
60         button_container.setFixedHeight(80) # Set fixed height for button area
61         button_container.setStyleSheet("""
62             QFrame {
63                 background-color: #34495e;
64                 border-radius: 10px;
65                 padding: 10px;
66             }
67         """)
68
69         # Button layout
70         button_layout = QHBoxLayout(button_container)
71         button_layout.setSpacing(20) # Space between buttons
72         button_layout.setContentsMargins(20, 0, 20, 0)
73
74         # Create and store buttons as instance variables
75         self.start_btn = QPushButton("Start Game")
76         self.landmark_btn = QPushButton("Toggle Landmarks")
77         self.reset_btn = QPushButton("Reset Game")
78         self.quit_btn = QPushButton("Quit")
79

```

```

80     # Store buttons in a list for easy access
81     self.buttons = [self.start_btn, self.landmark_btn, self.reset_btn, self.quit_btn]
82
83     button_style = """
84         QPushButton {
85             background-color: #2ecc71;
86             color: white;
87             border: none;
88             border-radius: 5px;
89             padding: 10px 20px;
90             font-weight: bold;
91             font-size: 14px;
92             min-width: 150px;
93         }
94         QPushButton:hover {
95             background-color: #27ae60;
96         }
97         QPushButton:pressed {
98             background-color: #1e8449;
99         }
100        QPushButton:disabled {
101            background-color: #95a5a6;
102        }
103    """
104
105    # Add all buttons to layout
106    for btn in self.buttons:
107        btn.setStyleSheet(button_style)
108        button_layout.addWidget(btn)
109
110    # Add button container to main layout
111    main_layout.addWidget(button_container)
112
113    # Connect buttons
114    self.start_btn.clicked.connect(self.start_game)
115    self.landmark_btn.clicked.connect(self.toggle_landmarks)
116    self.reset_btn.clicked.connect(self.reset_game)
117    self.quit_btn.clicked.connect(self.close)
118
119    # Initial button states
120    self.reset_btn.setEnabled(False)
121
122    # Window setup
123    self.setWindowTitle('Swipe That Face!')
124    self.setMinimumSize(1280, 820) # 720 + button area + margins
125    self.setStyleSheet("background-color: #2c3e50;") # Dark background
126
127    # Setup timer for video update
128    self.timer = QTimer()
129    self.timer.timeout.connect(self.update_frame)
130    self.timer.start(30) # 30ms = ~33fps
131
132    def start_game(self):
133        """Handle start button click"""
134        self.game.start_game()
135        self.start_btn.setEnabled(False)
136        self.reset_btn.setEnabled(False)
137
138    def update_frame(self):
139        if self.is_closing:
140            return
141
```

```

142     frame = self.game.get_current_frame()
143     if frame is not None:
144         h, w = frame.shape[:2]
145         q_img = QImage(frame.data, w, h, frame.strides[0], QImage.Format.Format_BGR888)
146         self.video_label.setPixmap(QPixmap.fromImage(q_img))
147
148         # Update button states only if window is still active
149         try:
150             if self.game.game_completed:
151                 self.reset_btn.setEnabled(True)
152                 self.start_btn.setEnabled(False)
153             elif self.game.game_state == self.game.MENU:
154                 self.start_btn.setEnabled(True)
155                 self.reset_btn.setEnabled(False)
156         except RuntimeError:
157             # Handle case where buttons might be deleted
158             pass
159
160     def toggle_landmarks(self):
161         self.game.show_landmarks = not self.game.show_landmarks
162
163     def reset_game(self):
164         """Handle reset button click"""
165         if self.game.game_completed:
166             self.game.reset_game()
167             self.game.game_state = self.game.MENU
168             self.start_btn.setEnabled(True)
169             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)`

Memalik status boolean `self.game.show_landmarks` yang digunakan untuk menyembunyikan/menampilkan 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 memproses gambar dan video, fungsinya mencakup membuka dan mengambil video dari camera (VideoCapture). OpenCV menjadi bagian yang penting dalam pemrosesan visual untuk deteksi dan menampilkan *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 mendeteksi titik-titik di wajah (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 bawaan 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 `facedetector`, `expression`, `gui`, `utils`, dan `game`. Contohnya, `expression.py` mengimpor `FaceDetector` dari `facedetector`, 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:

```
1 from PyQt6.QtWidgets import QApplication
2 from gui import launch_camera_selector
3 from qtgame import GameWindow
4 import sys
5
6 def main():
7     app = QApplication(sys.argv)
8     cam_index = launch_camera_selector()
9     if cam_index is not None:
10         window = GameWindow(cam_index)
11         window.show()
12         sys.exit(app.exec())
13
14 if __name__ == "__main__":
15     main()
```

Kode 1: Struktur Utama `main.py`

6.1.1 Deskripsi Kode

Kode ini melakukan langkah-langkah berikut:

1. Mengontrol Eksekusi:

- 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

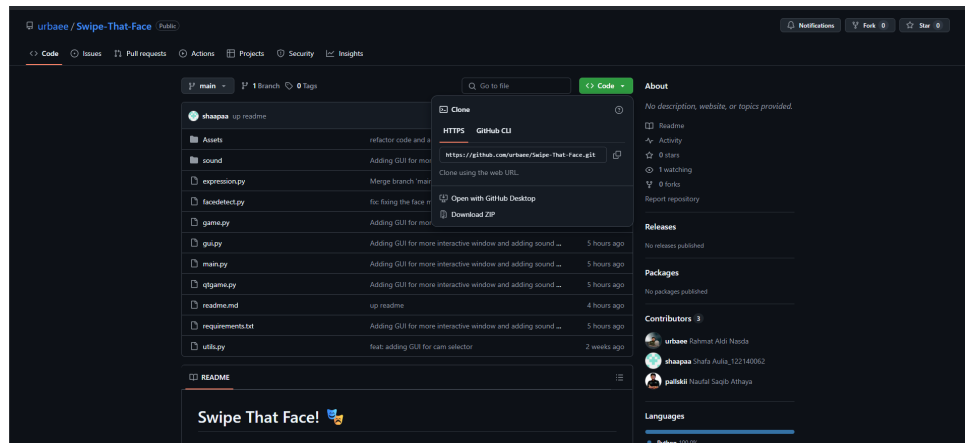
Langkah pertama dalam mengimplementasikan program ini adalah menyalin URL repository dari GitHub. Berikut adalah langkah-langkahnya:

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

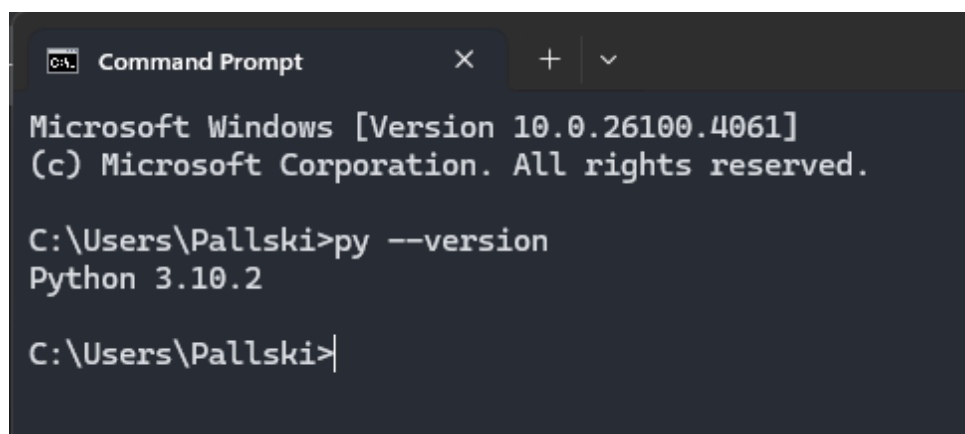
7.2 Langkah 2: Memeriksa Versi Python

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:

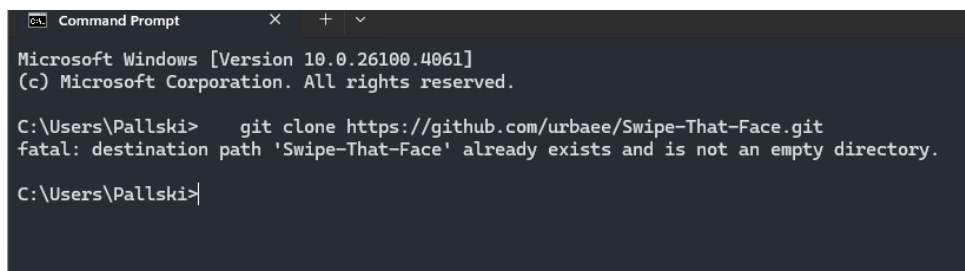
1. Buka terminal atau command prompt.
2. Pindah ke direktori tujuan untuk menyimpan repository dengan perintah:

```
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.

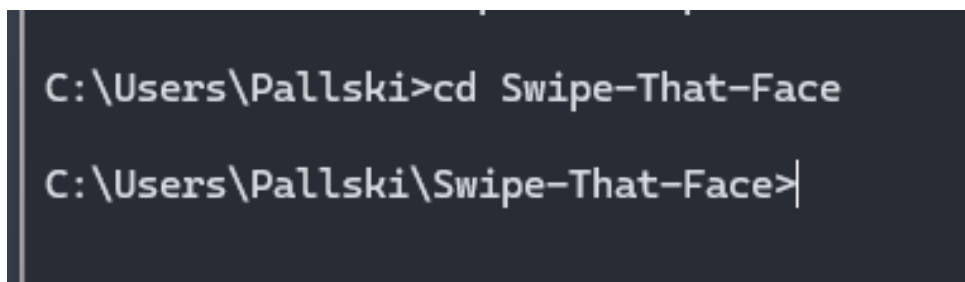


Gambar 3: Apply Clone Repository

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 environment seperti uv atau conda untuk menghindari tabrakan paket.

`uv pip install -r requirements.txt`

Pastikan semua library terinstal tanpa error sebelum melanjutkan.

```
C:\Users\pallaki\Swipe-That-Facepy -m pip install -r requirements.txt
Requirement already satisfied: absl-py==2.2.2 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 1)) (2.2.2)
Requirement already satisfied: attrs==25.0.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 2)) (25.0.0)
Requirement already satisfied: cffi==1.17.1 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 3)) (1.17.1)
Requirement already satisfied: contourpy==1.2.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 4)) (1.2.0)
Requirement already satisfied: cycler==0.12.1 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 5)) (0.12.1)
Requirement already satisfied: flatbuffers==25.2.10 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 6)) (25.2.10)
Requirement already satisfied: fonttools==4.57.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 7)) (4.57.0)
Requirement already satisfied: fsspec==2025.0.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 8)) (2025.0.0)
Requirement already satisfied: h5netcdf==1.4.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 9)) (1.4.0)
Requirement already satisfied: h5py==3.11.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 10)) (3.11.0)
Collecting mediapipe==0.10.0-cp310-cp310-win_amd64.whl (50.5 MB)
  Downloading mediapipe-0.10.0-cp310-cp310-win_amd64.whl (50.5 MB)
    90.5 MB 819 kB/s
Requirement already satisfied: ml-dtypes==0.5.1 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 13)) (0.5.1)
Requirement already satisfied: numpy==1.26.4 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 14)) (1.26.4)
Requirement already satisfied: opencv-contrib-python==4.10.0.86 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 15)) (4.10.0.86)
Requirement already satisfied: opencv-python==4.10.0.86 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 16)) (4.10.0.86)
Requirement already satisfied: ort==1.16.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 17)) (1.16.0)
Requirement already satisfied: packaging==25.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 18)) (25.0)
Collecting pandas==2.2.3
  Downloading pandas-2.2.3-cp310-cp310-win_amd64.whl (11.6 MB)
    11.6 MB 427 kB/s
Requirement already satisfied: pillow==11.2.1 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 20)) (11.2.1)
Collecting playsound==1.2.2
  Downloading playsound-1.2.2-py2.py3-none-any.whl (6.0 kB)
Collecting protobuf==5.29.3
  Downloading protobuf-5.29.3-cp310-cp310-win_amd64.whl (904 kB)
    904 kB 656 kB/s
Requirement already satisfied: pydantic==2.22 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 23)) (2.22)
Requirement already satisfied: pydantic-core==2.3 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 24)) (2.3)
Requirement already satisfied: pyqt==5.15.9 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 25)) (5.15.9)
Requirement already satisfied: pyqt6==6.9.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 26)) (6.9.0)
Requirement already satisfied: pyqt6-qt6==6.9.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 27)) (6.9.0)
Requirement already satisfied: python-dateutil==2.9.0.post0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 28)) (2.9.0.post0)
Collecting pyttsx3==2025.2
  Downloading pyttsx3-2025.2-py2.py3-none-any.whl (580 kB)
    580 kB 172 kB/s
Requirement already satisfied: scikit-learn==1.5.2 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 30)) (1.5.2)
Requirement already satisfied: sentencepiece==0.2.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 31)) (0.2.0)
Requirement already satisfied: six==1.17.0 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 32)) (1.17.0)
Requirement already satisfied: sounddevice==0.5.1 in c:\users\pallaki\appdata\local\programs\python\python310\lib\site-packages (from -r requirements.txt (line 33)) (0.5.1)
Collecting tdata==2025.2
  Downloading tdata-2025.2-py2.py3-none-any.whl (347 kB)
    347 kB 819 kB/s
Installing collected packages: tdata, pyttsx3, protobuf, playsound, pandas, mediapipe
Attempting uninstall: protobuf
Found existing installation: protobuf 4.25.7
Successfully uninstalled protobuf-4.25.7
```

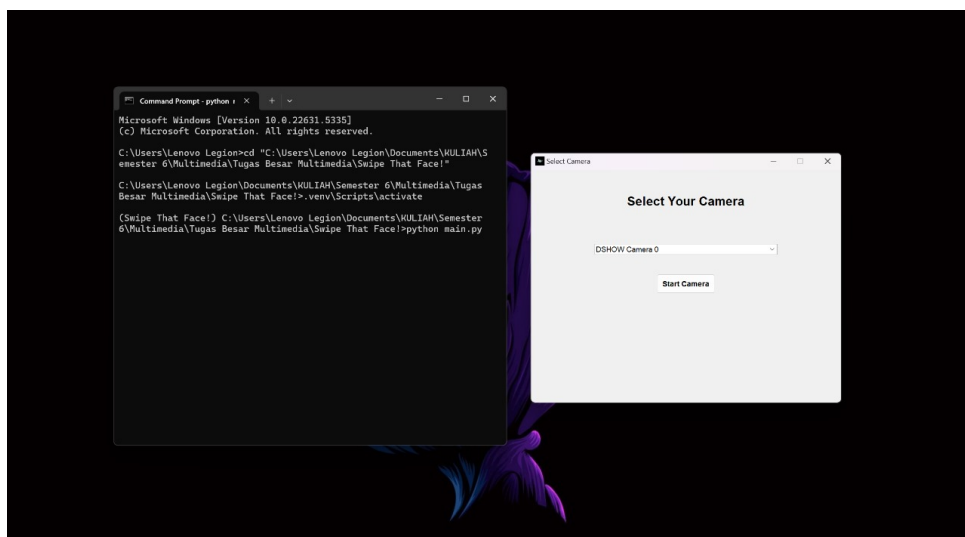
Gambar 5: Install Requirements

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:

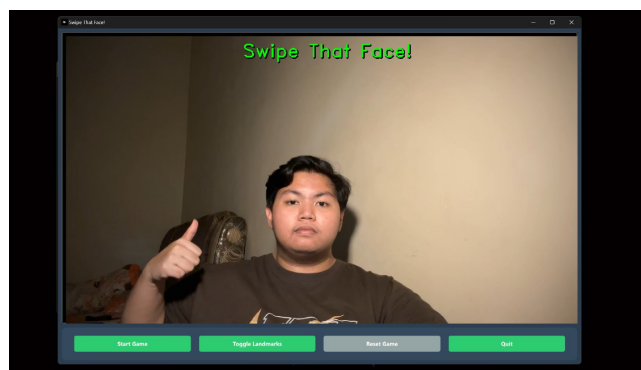
1. **Start Game:** Digunakan untuk memulai sesi permainan. Setelah ditekan, akan ada hitung mundur singkat sebelum tantangan ekspresi pertama ditampilkan.
2. **Toggle Landmarks:** Memungkinkan pengguna untuk menampilkan atau menyembunyikan titik-titik *landmark* wajah yang dideteksi oleh sistem pada 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:

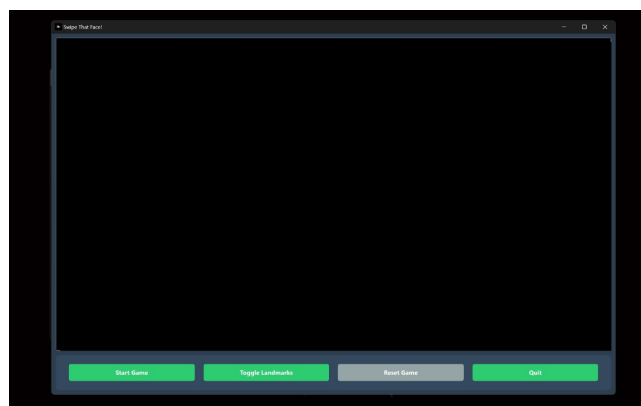
- **Start Game:**



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**

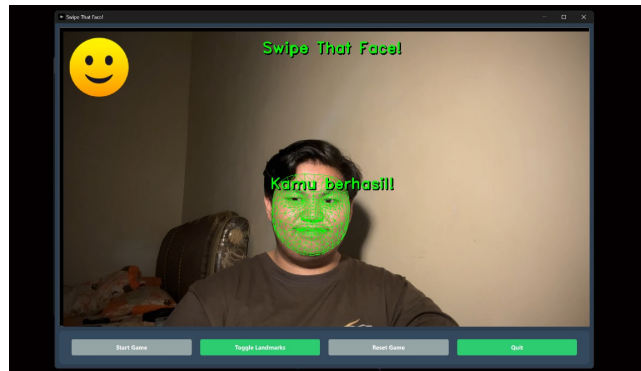
- **Preview:**



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.
- [8] R. C. Limited, “Introduction to pyqt,” 2025, available at: <https://www.riverbankcomputing.com/software/pyqt/intro/>.

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_iyg9z1TWJah6AAdnGph?usp=sharing

Lampiran

- **Link repository github:**
<https://github.com/urbaee/Swipe-That-Face>
- **Link youtube demo:**
<https://youtu.be/mJ7a0TqJscs>