Program Studi Teknik Informatika
Institut Teknologi Sumatera

Nama: **Bayu Ega Ferdana, Hamka Putra Andiyan, Falih Dzakwan Zuhdi** Tugas : **Final Project Report**
Mata Kuliah: **Sistem Teknologi Multimedia (IF25-40305)** Tanggal: 26 November 2025

1 Pendahuluan

1.1 Latar Belakang

Ekspresi wajah merupakan salah satu bentuk komunikasi non-verbal yang paling penting dalam interaksi manusia [1]. Dengan perkembangan teknologi Computer Vision dan Machine Learning, deteksi ekspresi wajah telah menjadi area penelitian yang menarik dengan berbagai aplikasi praktis.

Expressify adalah sebuah game interaktif yang memanfaatkan teknologi face detection dan expression recognition untuk menciptakan pengalaman bermain yang unik. Game ini menantang pemain untuk meniru berbagai ekspresi wajah (senang, sedih, kaget, dan datar) dalam batas waktu tertentu, dengan sistem penilaian otomatis berbasis landmark detection.

1.2 Tujuan Proyek

Proyek ini bertujuan untuk:

- Mengembangkan sistem deteksi ekspresi wajah real-time menggunakan MediaPipe Face Mesh [2] [3, 2]
- Menciptakan game interaktif yang edukatif dan menghibur
- Mengimplementasikan arsitektur modular untuk maintainability dan scalability
- Menyediakan user experience yang smooth dengan visual effects dan audio feedback

1.3 Fitur Utama

1. **Real-time Face Detection**: Deteksi wajah menggunakan 478 facial landmarks
2. **Expression Recognition**: Mendeteksi 4 ekspresi dasar (Happy, Sad, Surprised, Neutral)
3. **Multi-level Difficulty**: 3 tingkat kesulitan dengan durasi dan jumlah ekspresi berbeda
4. **Leaderboard System**: Top 10 scores untuk setiap difficulty level
5. **Modular UI Architecture**: 11 module terpisah untuk maintainability optimal
6. **Visual & Audio Effects**: Particle systems, animations, dan sound feedback

2 Arsitektur Sistem

2.1 Teknologi yang Digunakan

Proyek Expressify menggunakan beberapa teknologi utama sebagai berikut:

Tabel 1: Teknologi dan Versi

Teknologi	Versi	Fungsi
Python [4]	3.8+	Programming language utama
MediaPipe [3]	0.10.14	Face mesh detection dengan 478 landmarks
OpenCV [5]	4.10.0.84	Computer vision dan camera capture
Pygame [6]	2.6.0	Game engine dan UI rendering
NumPy [7]	1.26.4	Numerical computation untuk landmark analysis
Pillow	10.4.0	Image processing dan manipulation

2.2 Struktur Project

Project Expressify menggunakan arsitektur modular dengan struktur sebagai berikut:

```
Expressify/
|-- src/                    # Source code utama
|   |-- main.py             # Entry point & game controller
|   |-- face_detector.py    # MediaPipe face detection
|   |-- game_logic.py       # Game rules & scoring
|   |-- sound_manager.py    # Audio system
|   |-- leaderboard_manager.py  # Score persistence
|   +-- ui/                 # Modular UI components (11 modules)
|-- assets/                 # Game assets (images, sounds, photos)
|-- docs/                   # Documentation
|-- reports/                # Project reports (LaTeX)
+-- requirements.txt        # Python dependencies
```

Kode 1: Struktur Folder Utama

2.3 Arsitektur UI Modular

Salah satu keunggulan Expressify adalah arsitektur UI yang modular. UI manager yang awalnya monolithic (650+ lines) telah di-refactor menjadi 11 module terpisah untuk meningkatkan maintainability:

- **ui_manager.py**: Main orchestrator yang mengkoordinasikan semua UI components
- **constants.py**: Configuration (Colors, Dimensions, FontManager)
- **base_renderer.py**: Reusable rendering utilities (gradients, text effects)
- **animations.py**: Animation systems (ParticleSystem, FloatingImages, Confetti)
- **image_manager.py**: Expression image loading & rendering
- **menu_screen.py**: Main menu screen renderer
- **game_screen.py**: Game playing screen renderer
- **results_screen.py**: Results & ranking screen renderer
- **other_screens.py**: Additional screens (Difficulty, Leaderboard, NameInput)

3 Implementasi

3.1 Face Detection dengan MediaPipe

Sistem deteksi wajah menggunakan MediaPipe Face Mesh yang menyediakan 478 facial landmarks. Berikut adalah implementasi deteksi ekspresi:

```python
class FaceDetector:
    def __init__(self):
        self.mp_face_mesh = mp.solutions.face_mesh
        self.face_mesh = self.mp_face_mesh.FaceMesh(
            max_num_faces=1,
            refine_landmarks=True,
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5
        )

    def detect_expression(self, landmarks):
        # Mouth landmarks untuk deteksi ekspresi
        mouth_top = landmarks[13].y
        mouth_bottom = landmarks[14].y
        mouth_left = landmarks[61].x
        mouth_right = landmarks[291].x

        # Eye landmarks untuk deteksi kaget
        left_eye_top = landmarks[159].y
        left_eye_bottom = landmarks[145].y

        # Hitung mouth openness dan aspect ratio
        mouth_height = abs(mouth_bottom - mouth_top)
        mouth_width = abs(mouth_right - mouth_left)
        mouth_aspect_ratio = mouth_height / mouth_width

        # Deteksi ekspresi berdasarkan geometri
        if mouth_aspect_ratio > 0.35:
            return "surprised"  # Mulut terbuka lebar
        elif mouth_top < mouth_bottom - 0.02:
            return "happy"      # Sudut mulut naik
        elif mouth_top > mouth_bottom + 0.01:
            return "sad"        # Sudut mulut turun
        else:
            return "neutral"    # Wajah rileks
```

Kode 2: Face Detector Implementation

3.2 Game Logic dan Scoring System

Game logic mengatur flow permainan dan sistem penilaian:

```python
class GameLogic:
    DIFFICULTY_SETTINGS = {
        "Mudah": {"duration": 30, "expressions": 2},
        "Menengah": {"duration": 20, "expressions": 4},
        "Sulit": {"duration": 15, "expressions": 4}
    }

    def calculate_score(self, correct_count, total_count):
        """Calculate score percentage"""
        if total_count == 0:
            return 0
        return int((correct_count / total_count) * 100)

```

```
    def get_rank(self, percentage):
        """Determine rank based on score percentage"""
        if percentage >= 80:
            return "S"  # Luar Biasa
        elif percentage >= 60:
            return "A"  # Bagus Sekali
        elif percentage >= 40:
            return "B"  # Cukup Baik
        else:
            return "C"  # Terus Berlatih
```

Kode 3: Game Logic Core

3.3 Modular UI Architecture

Implementasi modular UI memisahkan concerns dan meningkatkan reusability:

```
class UIManager:
    def __init__(self, screen):
        self.screen = screen

        # Initialize all subsystems
        self.menu_screen = MenuScreen(screen)
        self.game_screen = GameScreen(screen)
        self.results_screen = ResultsScreen(screen)
        self.difficulty_screen = DifficultyScreen(screen)
        self.leaderboard_screen = LeaderboardScreen(screen)
        self.name_input_screen = NameInputScreen(screen)

        # Animation systems
        self.particle_system = ParticleSystem()
        self.floating_image_system = FloatingImageSystem()
        self.confetti_system = ConfettiSystem()

    def draw(self, state, game_data):
        """Delegate rendering to appropriate screen"""
        if state == "menu":
            self.menu_screen.draw()
        elif state == "game":
            camera_area = self.game_screen.get_camera_area()
            self.particle_system.update_and_draw(
                self.screen,
                exclude_area=camera_area
            )
            self.game_screen.draw(game_data)
        # ... other states
```

Kode 4: UI Manager Orchestration

3.4 Visual Effects dan Animations

Sistem animasi menciptakan pengalaman visual yang menarik:

```
class ParticleSystem:
    def __init__(self):
        self.particles = []

    def add_particle(self, x, y, color):
        self.particles.append({
            'x': x, 'y': y,
            'vx': random.uniform(-2, 2),
            'vy': random.uniform(-3, -1),
```

```
            'color': color,
            'life': 60,  # frames
            'size': random.randint(3, 8)
        })

    def update_and_draw(self, screen, exclude_area=None):
        for particle in self.particles[:]:
            # Update position
            particle['x'] += particle['vx']
            particle['y'] += particle['vy']
            particle['vy'] += 0.2  # gravity
            particle['life'] -= 1

            # Check exclusion zone (avoid camera area)
            if exclude_area and self.in_exclusion_zone(
                particle, exclude_area
            ):
                continue

            # Draw particle
            alpha = int(255 * (particle['life'] / 60))
            pygame.draw.circle(screen, particle['color'],
                (int(particle['x']), int(particle['y'])),
                particle['size'])
```

Kode 5: Particle System Implementation

4 Fitur-Fitur Game

4.1 Sistem Difficulty

Game menyediakan 3 tingkat kesulitan dengan karakteristik berbeda:

Tabel 2: Pengaturan Difficulty Level

Difficulty	Durasi (s)	Jumlah Ekspresi	Challenge
Mudah	30	2	Waktu santai
Menengah	20	4	Balanced challenge
Sulit	15	4	Time pressure

4.2 Sistem Ranking dan Leaderboard

Sistem ranking menggunakan grade S, A, B, C dengan visual yang menarik:

- **Rank S** ($\geq 80\%$): "LUAR BIASA!" - Gold color dengan 5 bintang
- **Rank A** (60-79%): "BAGUS SEKALI!" - Silver color dengan 4 bintang
- **Rank B** (40-59%): "CUKUP BAIK!" - Bronze color dengan 3 bintang
- **Rank C** ($< 40\%$): "TERUS BERLATIH!" - Gray color dengan 2 bintang

Leaderboard menyimpan top 10 scores untuk setiap difficulty level dengan format:

```
{
    "Mudah": [
        {"name": "Player1", "score": 95, "rank": "S"},
        {"name": "Player2", "score": 85, "rank": "S"},
```

```
        ...
    ],
    "Menengah": [...],
    "Sulit": [...]
}
```

Kode 6: Leaderboard Data Structure

4.3 Auto-Replay Feature

Setelah game selesai, pemain dapat:

- Tekan **SPACE**: Langsung ke difficulty selection dengan nama tersimpan
- Tekan **ESC**: Kembali ke menu utama (nama di-reset)

Fitur ini meningkatkan user experience dengan mengurangi repetitive input.

4.4 Exclusion Zones

Untuk menghindari overlap visual antara camera feed dengan UI elements, implementasi exclusion zones diterapkan:

- **Particle Exclusion**: 30px margin dari camera area
- **Floating Images Exclusion**: 50px margin dari camera area
- **Rounded Camera Corners**: Border radius 12px untuk tampilan modern

5 Testing dan Quality Assurance

5.1 Testing Methodology

Testing dilakukan secara manual dengan fokus pada:

1. **Functional Testing**: Memastikan semua fitur berjalan sesuai ekspektasi
2. **Usability Testing**: Menguji intuitivitas controls dan UI

5.2 Known Limitations

Beberapa limitasi yang teridentifikasi:

- Akurasi deteksi sangat bergantung pada kondisi pencahayaan
- Memerlukan webcam dengan resolusi minimal 640x480
- Performa optimal pada single face detection
- Ekspresi ekstrim dapat menghasilkan false detection

5.3 Future Improvements

Rencana pengembangan ke depan:

- Menambah variasi ekspresi (marah, takut, jijik)
- Multiplayer mode dengan competitive scoring
- Cross-platform deployment (mobile, web)
- Cloud-based global leaderboard

6 Distribusi dan Deployment

6.1 Package Requirements

Expressify menggunakan beberapa dependencies utama yang harus di-package dalam executable:

- **MediaPipe** (0.10.0+): Face mesh detection dengan binary data files (.binarypb)
- **OpenCV** (cv2): Image processing dan camera capture
- **Pygame** (2.5.0+): Game engine, rendering, dan audio
- **NumPy**: Numerical operations untuk landmark calculations
- **Matplotlib**: Visualisasi data (hidden dependency)
- **Pillow (PIL)**: Image loading dan processing

6.2 PyInstaller Configuration

Untuk menghasilkan executable Windows (.exe), digunakan PyInstaller dengan konfigurasi khusus untuk mengatasi berbagai dependency issues:

```
# Expressify.spec
a = Analysis(
    ['src/main.py'],
    pathex=[],
    binaries=[],
    datas=[
        ('assets', 'assets'),
        ('src/ui', 'src/ui'),
    ],
    hiddenimports=[
        'mediapipe', 'cv2', 'pygame', 'numpy',
        'PIL', 'matplotlib', 'matplotlib.pyplot'
    ],
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=None,
    noarchive=False,
)

# Collect MediaPipe binary data files
a.datas += collect_data_files('mediapipe')
```

Kode 7: PyInstaller Spec Configuration

6.3 Asset Path Compatibility

Challenge utama dalam PyInstaller adalah path resolution untuk assets (images, sounds). Solusinya adalah implementasi helper function yang mendeteksi frozen state:

```
def get_base_path():
    """Get base path for assets (works in dev and frozen mode)"""
    if getattr(sys, 'frozen', False):
        # Running as compiled executable
        return sys._MEIPASS
```

```
    else:
        # Running in development
        return os.path.dirname(os.path.abspath(__file__))

# Usage in sound_manager.py
ASSETS_DIR = os.path.join(get_base_path(), "assets", "sounds")

# Usage in image_manager.py
base_path = os.path.join(get_base_path(), "assets", "photo")
```

Kode 8: Asset Path Resolution

Function ini menggunakan `sys._MEIPASS` - temporary folder yang dibuat PyInstaller saat runtime untuk extract assets dari executable bundle.

6.4 Build Scripts

Untuk mempermudah build process, dibuat batch scripts dengan dua mode distribusi:

Portable Version (–onefile):

- Single executable file (~150-200 MB)
- Slower startup (extract ke temp folder)
- Ideal untuk quick distribution

Install Version (–onedir):

- Folder berisi executable + dependencies (~180 MB)
- Faster startup (no extraction needed)
- Professional installation experience

```
pyinstaller ^
    --name="Expressify" ^
    --onefile ^
    --windowed ^
    --icon="assets/images/icon.ico" ^
    --add-data="assets;assets" ^
    --add-data="src/ui;src/ui" ^
    --hidden-import="mediapipe" ^
    --collect-data="mediapipe" ^
    --noconsole ^
    src/main.py
```

Kode 9: Build Script (build_both.bat)

6.5 Icon Generation

Application icon dibuat dari emoji asset (Senang.png) menggunakan PIL:

```
from PIL import Image

img = Image.open("assets/photo/Senang.png")
icon_sizes = [(256,256), (128,128), (64,64),
              (48,48), (32,32), (16,16)]
img.save("assets/images/icon.ico",
         format='ICO', sizes=icon_sizes)
```

Kode 10: Icon Generation Script

Icon ini muncul di taskbar, file explorer, window title bar, dan desktop shortcuts, memberikan identitas visual yang konsisten untuk aplikasi.

6.6 Deployment Challenges & Solutions

Tabel 3: PyInstaller Challenges dan Solutions

Challenge	Solution
MediaPipe .binarypb files missing	Add `collect_data_files('mediapipe')`
Matplotlib not found	Add to `hiddenimports` list
Assets not loading in exe	Implement `get_base_path()` with `sys._MEIPASS`
Large file size (200+ MB)	Expected due to MediaPipe + OpenCV binaries
Slow startup (onefile)	Offer onedir version for better performance

6.7 Distribution Strategy

Final distribution melalui GitHub Releases dengan dua files:

1. **Expressify-Portable.exe**: Single file untuk quick download
2. **Expressify-Install.zip**: Folder version dengan faster startup

Kedua versi fully functional tanpa memerlukan Python installation, webcam drivers otomatis terdeteksi oleh OpenCV, dan game siap dimainkan setelah download.

7 Kesimpulan

Expressify berhasil mengimplementasikan sistem facial expression recognition dalam bentuk game interaktif yang edukatif dan menghibur. Proyek ini mendemonstrasikan beberapa pencapaian penting:

1. **Technical Achievement**: Integrasi sukses antara MediaPipe, OpenCV, dan Pygame untuk real-time face detection dan game rendering
2. **Architectural Excellence**: Refactoring dari monolithic code (650+ lines) ke modular architecture (11 modules) yang meningkatkan maintainability dan scalability
3. **User Experience**: Implementasi visual effects (particles, animations, rounded corners) dan audio feedback menciptakan pengalaman bermain yang engaging
4. **Code Quality**: Clean code practices dengan separation of concerns, reusable components, dan clear dependencies
5. **Feature Completeness**: Sistem ranking, leaderboard, multiple difficulty levels, dan auto-replay feature membuat game lebih kompetitif dan replayable

Proyek ini membuktikan bahwa computer vision technology dapat diintegrasikan dengan game development untuk menciptakan aplikasi yang tidak hanya teknis kompleks, namun juga fun dan user-friendly. Arsitektur modular yang diterapkan memungkinkan pengembangan future features dengan effort minimal.

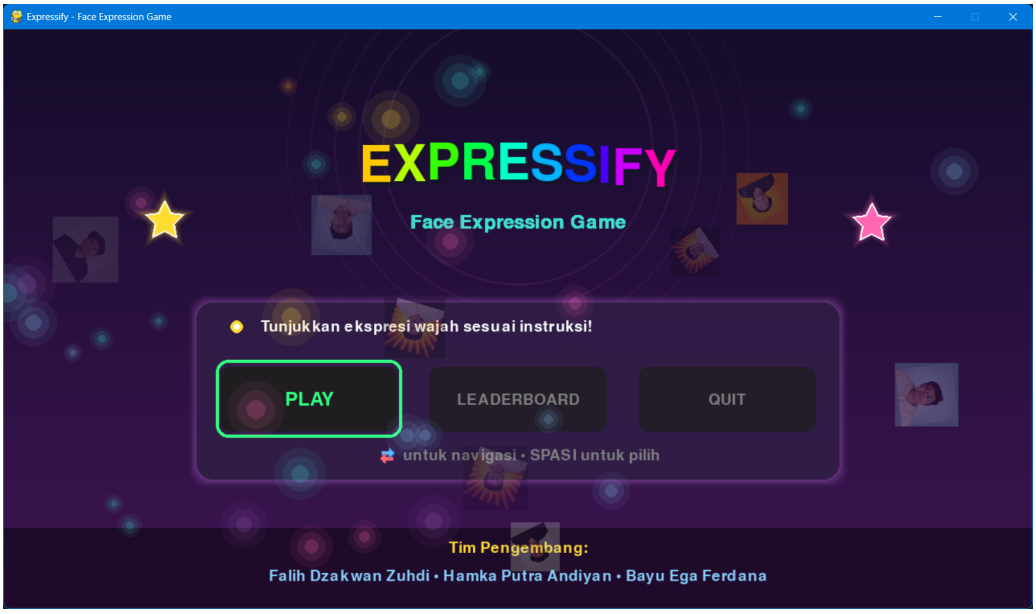
Lessons Learned:

- Modular architecture sangat penting untuk long-term maintainability
- User experience details (animations, exclusion zones) membuat perbedaan signifikan

- Testing iteratif membantu mengidentifikasi edge cases dalam face detection
- Documentation dan clean code mempermudah collaboration dalam tim

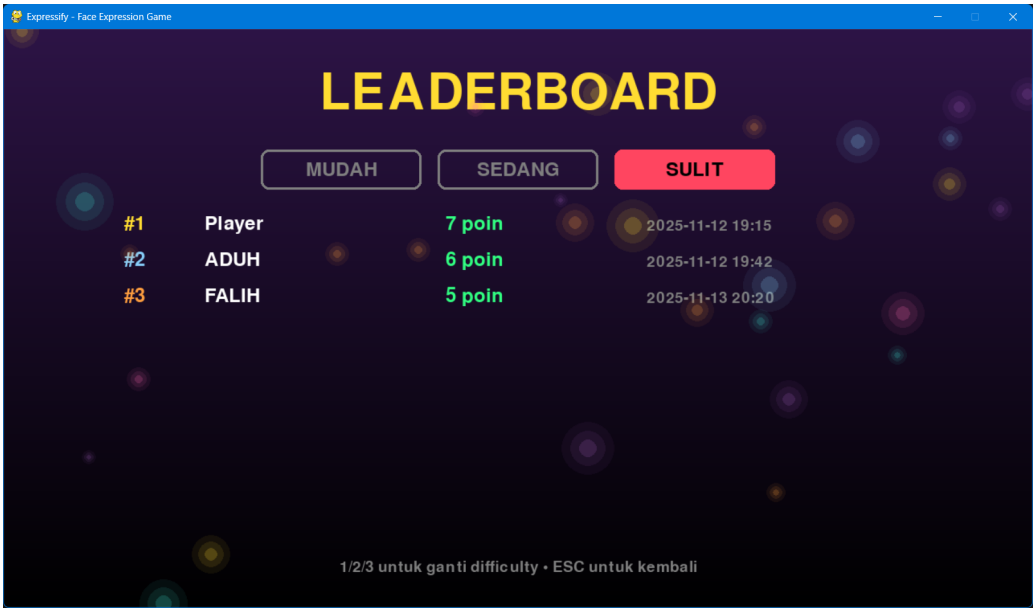
Expressify adalah contoh sukses bagaimana teknologi AI dan Computer Vision dapat dikemas dalam format yang accessible dan entertaining untuk end users, sekaligus maintaining technical excellence dan code quality standards.

Lampiran: Screenshots



Gambar 1: Main Menu

Gambar 2: Game Screen dengan Real-time Face Detection



Gambar 3: Leaderboard Screen

References

[1] P. Ekman, "An argument for basic emotions," *Cognition & emotion*, vol. 6, no. 3-4, pp. 169–200, 1992.

[2] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee *et al.*, "Mediapipe: A framework for building perception pipelines," in *arXiv preprint arXiv:1906.08172*, 2019.

[3] Google LLC, "MediaPipe," https://developers.google.com/mediapipe, 2023, accessed: 2025-11-26.

[4] Python Software Foundation, "Python 3 Documentation," https://docs.python.org/3/, 2024, accessed: 2025-11-26.

[5] OpenCV Team, "OpenCV: Open Source Computer Vision Library," https://opencv.org/, 2024, accessed: 2025-11-26.

[6] Pygame Community, "Pygame: Python Game Development," https://www.pygame.org/, 2024, accessed: 2025-11-26.

[7] NumPy Developers, "NumPy: The fundamental package for scientific computing with Python," https://numpy.org/, 2024, accessed: 2025-11-26.