🚀 Quick Start Guide - Face Recognition WebSocket Client

📦 File yang Telah Dibuat

1. **web_face_auth.html** - Interface web HTML dengan WebSocket client
2. **web_server.py** - Python HTTP server untuk melayani HTML
3. **web_config.json** - File konfigurasi dengan credentials dan settings
4. **run_web_server.sh** - Helper script untuk run server
5. **WEB_CLIENT_README.md** - Dokumentasi lengkap
6. **web_client_config.json** - Config template

⚡ Quick Start - RECOMMENDED METHOD ⭐

Method 1: Python CLI (✅ WORKS RELIABLY)

Ini adalah cara yang paling reliable! WebSocket connection works perfectly tanpa browser limitations.

```
# Menggunakan config profile (paling mudah)
python test_websocket_auth.py --profile production enrollment 653384

# Atau dengan credentials langsung
python test_websocket_auth.py \
  frapi_YY7OEJn1FCyDoGiGLwiueTw79hkQWduGNy2L-XbsCB4 \

_lwZfcqmdsi5PtRLjmOeTDgTxP5JaRAN3r4i6IpCSOC6ndL536sO9ZuFVjbgLshbmuNKmBBu
ty_wZgdyXEw-DA \
  https://face.ahu.go.id \
  enrollment \
  653384
```

Keuntungan:

- ✅ WebSocket connection **always works**
- ✅ No CORS issues
- ✅ No browser security restrictions
- ✅ Better performance
- ✅ Full error messages
- ✅ Perfect untuk production use

Method 2: Web Interface (⚠ WebSocket May Fail Due to CORS)

Web interface bagus untuk visual feedback, tapi WebSocket dari browser sering di-block oleh CORS policy.

Status:

- ✅ HTTPS API calls: **Working**
- ❌ WebSocket connection: **May fail due to browser CORS**

Jika WebSocket gagal, gunakan **Method 1 (CLI)** di atas.

Langkah 1: Jalankan Web Server

```
# Cara paling mudah
python web_server.py

# Atau menggunakan bash script
./run_web_server.sh

# Atau dengan options
python web_server.py --port 3000
python web_server.py --public --port 8080  # untuk network access
```

Server akan running di: **http://localhost:8080**

Langkah 2: Buka Browser

Buka browser dan akses: **http://localhost:8080**

Langkah 3: Pilih Profile & Start

1. **Pilih Profile**: Production (sudah ter-config dengan credentials Anda)
2. **Pilih Mode**: Enrollment atau Authentication
3. **Masukkan User ID**: Contoh: `653384`
4. **Click**: "Connect & Start"

🔀 CLI vs Web Interface

Python CLI Client ⭐ RECOMMENDED

Status: ✅ Fully Working

```
python test_websocket_auth.py --profile production enrollment 653384
```

Pros:

- ✅ WebSocket connection always works
- ✅ No CORS/browser security issues
- ✅ Reliable and stable
- ✅ Full control and debugging
- ✅ Better error messages

- ✅ Production ready
- ✅ Real-time visual feedback (OpenCV window)
- ✅ Face mesh landmarks visualization
- ✅ Liveness detection indicators

Cons:

- ⚠ Requires Python + OpenCV + dependencies
- ⚠ Requires webcam access setup

Best for:

- Production use
- Reliable testing
- Automation/scripting
- Backend integration
- Situations yang butuh stability

Web Browser Interface

Status: ⚠ Partial - API works, WebSocket may fail

```
python web_server.py
# Open: http://localhost:8080
```

Pros:

- ✅ No Python dependencies needed
- ✅ Easy to access (just browser)
- ✅ Modern UI
- ✅ Easy for demos

Cons:

- ❌ WebSocket blocked by browser CORS policy
- ❌ Connection fails on production server
- ❌ Limited to browser security restrictions

Best for:

- Quick demos (if WebSocket works)
- UI preview
- Local development server (if CORS configured)

🎯 Why CLI Works But Browser Doesn't?

Python CLI:

```
Python → Direct WebSocket → Server ✅
(No browser security, no CORS restrictions)
```

Browser:

```
Browser → CORS Check → ❌ BLOCKED
(Browser enforces same-origin policy)
```

Server Response:

- HTTPS API endpoints: ✅ CORS allowed
- WebSocket connections: ❌ CORS not configured for browser origin

Solution: Server perlu configure CORS headers untuk WebSocket, ATAU gunakan Python CLI (recommended).

🔧 Konfigurasi (web_config.json)

File `web_config.json` sudah dikonfigurasi dengan:

```
{
  "default_profile": "production",
  "profiles": {
    "production": {
      "base_url": "https://face.ahu.go.id",
      "api_key": "frapi_YY7OEJn1FCyDoGiGLwiueTw79hkQWduGNy2L-XbsCB4",
      "secret_key":
"_lwZfcqmdsi5PtRLjmOeTDgTxP5JaRAN3r4i6IpCSOC6ndL536sO9ZuFVjbgLshbmuNKmBB
uty_wZgdyXEw-DA"
    }
  }
}
```

Profile ini akan **auto-load** saat web interface dibuka!

💻 Command Line dengan Config

Script `test_websocket_auth.py` sekarang juga support config file:

Menggunakan Config Profile

```
# Enrollment dengan profile
python test_websocket_auth.py --profile production enrollment 653384
```

```
# Authentication dengan profile
python test_websocket_auth.py --profile production authentication 653384

# Identification (tanpa user_id)
python test_websocket_auth.py --profile production authentication
```

Menggunakan Direct Credentials (seperti sebelumnya)

```
python test_websocket_auth.py \
  frapi_YY7OEJn1FCyDoGiGLwiueTw79hkQWduGNy2L-XbsCB4 \

_lwZfcqmdsi5PtRLjmOeTDgTxP5JaRAN3r4i6IpCSOC6ndL536sO9ZuFVjbgLshbmuNKmBBu
ty_wZgdyXEw-DA \
  https://face.ahu.go.id \
  enrollment \
  653384
```

🎯 Fitur Web Interface

Visual Feedback Real-Time

- ✅ Face bounding box dengan corner markers
- ✅ Face mesh landmarks (oval, eyebrows, nose, lips)
- ✅ Eye regions untuk blink detection
- ✅ Status panel dengan live metrics
- ✅ Log output dengan color coding

Liveness Detection Indicators

- **Blinks**: Counter dengan status [OK] atau [--]
- **EAR** (Eye Aspect Ratio): Value dengan bar indicator
- **Motion**: Status dengan counter [OK] atau [??]
- **Quality**: Score dengan threshold indicator
- **Liveness**: Score dengan verification status

Session Monitoring

- Connection status
- Session token
- Frames processed counter
- Real-time liveness score
- Blink count
- Motion events
- Quality score

Modals & Notifications

- ✅ Success modal untuk enrollment/authentication berhasil
- ❌ Error modal untuk troubleshooting
- ⚠ Obstacle warnings saat terdeteksi
- 📊 Similarity score dengan old photo (enrollment)

🔐 Modes

1. Enrollment (📝)

Mendaftarkan wajah user baru

Required:

- User ID

Optional:

- Old Profile Photo (untuk similarity comparison)

Process:

1. Capture multiple frames (target: 3 samples)
2. Validate face quality
3. Detect liveness (blinks + motion)
4. Check for obstacles (mask, glasses, dll)
5. Compare dengan old photo (jika ada)
6. Save face embeddings

2. Authentication (✓)

A. Verification (dengan user_id)

- Cocokkan dengan user tertentu
- Return: authenticated + confidence

B. Identification (tanpa user_id)

- Cari match dari semua users
- Return: user_id + confidence

Process:

1. Capture frames (min: 10 frames)
2. Validate liveness
3. Extract face embedding
4. Match dengan database
5. Return result + confidence

📂 File Structure

```
face_regocnition_v2/
├── web_face_auth.html          # Web interface
├── web_server.py               # HTTP server
├── web_config.json             # Configuration file ⭐
├── run_web_server.sh           # Helper script
├── test_websocket_auth.py      # CLI client (updated)
├── WEB_CLIENT_README.md        # Full documentation
├── web_client_config.json      # Config template
└── QUICK_START.md              # This file
```

🌐 Access dari Network Lain

Jika ingin access dari device lain di network yang sama:

```
# 1. Run server dengan --public
python web_server.py --public --port 8080

# 2. Cek IP address Anda
ifconfig | grep "inet "  # macOS/Linux
ipconfig                 # Windows

# 3. Buka dari device lain
http://<your-ip-address>:8080
```

⚙️ Settings di Config File

Anda bisa customize di `web_config.json`:

```
{
  "settings": {
    "camera": {
      "default_device": 0,
      "frame_rate": 10,
      "jpeg_quality": 80
    },
    "session": {
      "enrollment": {
        "target_samples": 3,
        "duration": 30
      },
      "authentication": {
        "min_frames_required": 10,
        "required_blinks": 1,
        "duration": 30
      }
    }
```

```
  }
}
```

🔒 Security Notes

⚠ Untuk Development Only!

- Config file berisi credentials dalam plaintext
- Tidak ada encryption untuk communication dengan server
- Tidak ada authentication untuk web server
- CORS enabled untuk semua origins

Untuk Production:

- Gunakan HTTPS/WSS
- Implement proper authentication
- Encrypt credentials
- Restrict CORS
- Add rate limiting
- Use environment variables untuk secrets

🪲 Troubleshooting

Web server tidak start

```
# Check port availability
lsof -i :8080

# Try different port
python web_server.py --port 3000
```

Camera tidak terdeteksi

- Allow camera permission di browser
- Check browser console (F12)
- Try different browser (Chrome recommended)

Config tidak ter-load

- Ensure `web_config.json` ada di same directory
- Check JSON syntax validity
- Restart web server

WebSocket connection failed

- Verify base URL di config
- Check API credentials

- Ensure API server running
- Check browser console untuk error details

Frame rejected / obstacles

- Ensure adequate lighting
- Remove glasses, mask, atau obstacles
- Keep face centered dan visible
- Blink naturally

📞 Next Steps

⭐ RECOMMENDED: Start dengan CLI

```
# 1. Test enrollment
python test_websocket_auth.py --profile production enrollment 653384

# 2. Test authentication (verification)
python test_websocket_auth.py --profile production authentication 653384

# 3. Test authentication (identification - tanpa user_id)
python test_websocket_auth.py --profile production authentication
```

Alternative: Try Web Interface

Jika ingin coba web interface (mungkin WebSocket gagal):

```
# 1. Start web server
python web_server.py

# 2. Open browser
http://localhost:8080

# 3. Jika WebSocket gagal, kembali ke CLI method
```

Production Deployment

Untuk production, **always gunakan CLI client**:

```
# Integration example
python test_websocket_auth.py --profile production enrollment
"${USER_ID}"
```

atau integrate langsung di Python code Anda dengan import class `FaceAuthWebSocketClient`.

📚 Dokumentasi Lengkap

Baca file `WEB_CLIENT_README.md` untuk dokumentasi lengkap tentang:

- API endpoints detail
- WebSocket message format
- Visual feedback customization
- Advanced configuration
- Integration guide

✅ Summary

Sekarang Anda punya 2 cara untuk testing:

🥇 Method 1: Python CLI (RECOMMENDED & WORKING)

```
python test_websocket_auth.py --profile production enrollment 653384
```

Status: ✅ Fully Working

- WebSocket connection: ✅ Works
- Visual feedback: ✅ OpenCV window with overlays
- Liveness detection: ✅ Full support
- Production ready: ✅ Yes

🥈 Method 2: Web Browser

```
python web_server.py  # Then open http://localhost:8080
```

Status: ⚠ Partial

- API calls: ✅ Works
- WebSocket: ❌ Blocked by CORS
- Use case: Demo/preview only

🎯 TL;DR - Quick Command

Paling mudah dan reliable:

```
python test_websocket_auth.py --profile production enrollment 653384
```

✅ No setup needed (config already loaded from web_config.json)
✅ WebSocket works perfectly

✅ Real-time visual feedback
✅ Blink & motion detection
✅ Face mesh landmarks
✅ Quality monitoring

That's it! 🎉

📚 Dokumentasi Lengkap

- **WEBSOCKET_TROUBLESHOOTING.md** - Penjelasan detail kenapa browser gagal
- **WEB_CLIENT_README.md** - Full documentation untuk web interface
- **Test CLI** - Sudah bisa langsung digunakan dengan `--profile production`

Enjoy! 🚀