LAMPIRAN

Kode Program:

Lampiran 1 Source Code import library

```
import os
import sys
import logging
import argparse
import tempfile
import time
import uuid
import threading
import requests
  import json
  from datetime import datetime
  from flask import Flask, request, jsonify
  from flask_cors import CORS
  import cv2
  import numpy as np
  from config import Config
  from face_detector_simple import FaceDetector
  from facenet_model_casia import FaceNetModel
```

Lampiran 2 Source Code konfigurasi laravel server

```
LARAVEL_CONFIG = {
'base_url': 'https://portaltelkom.my.id',  # URL Laravel
server
'register_endpoint': '/api/face-services/register-url',
'heartbeat_endpoint': '/api/face-services/heartbeat',
'log_endpoint': '/api/face-services/log',
'api_key': 'your-secret-facenet-key',  # Optional: untuk
security
}
```

Lampiran 3 Source Code Class untuk integrasi dengan Laravel server

```
class LaravelIntegration:
def __init__(self, config):
self.config = config
self.base_url = config['base_url']
self.register_endpoint = config['register_endpoint']
self.heartbeat_endpoint = config['heartbeat_endpoint']
self.log_endpoint = config['log_endpoint']
self.api_key = config.get('api_key')
def get_headers(self):
  """Get headers untuk request ke Laravel"""
  headers = {
  'Content-Type': 'application/json',
  'User-Agent': 'FaceNet-Server/1.0'
  }
  if self.api_key:
  headers['Authorization'] = f'Bearer {self.api_key}'
  return headers
```

```
18.   def register_facenet_url(self, ngrok_url,
   local_url='http://localhost:5001', facenet_model=None):
19.   """Register FaceNet URL ke Laravel server"""
20.   try:
21.   url = self.base_url + self.register_endpoint
22.   server_info = {
23.   'model_loaded': facenet_model is not None,
24.   'registered_identities':
   len(facenet_model.get_registered_identities()) if
   facenet_model else 0,
25.   'timestamp': datetime.now().isoformat(),
26.   'service_type': 'facenet'
27.   }
28.   if facenet_model:
29.   identities = facenet_model.get_registered_identities()
30.   server_info['identities'] = identities
31.   payload = {
32.   'service_name': 'facenet_verification',
33.   'ngrok_url': ngrok_url,
34.   'local_url': local_url,
35.   'server_info': server_info
36.   }
37.   response = requests.post(
38.   url,
39.   json=payload,
40.   headers=self.get_headers(),
41.   timeout=10
42.   )
43.   if response.status_code == 200:
44.   result = response.json()
45.   logger.info(f"✓ Successfully registered FaceNet to Laravel:
   {ngrok_url}")
46.   return True, result
47.   else:
48.   logger.error(f"❌ Failed to register FaceNet to Laravel:
   {response.status_code}")
49.   logger.error(f"Response: {response.text}")
50.   return False, response.text


51.   except Exception as e:
52.   logger.error(f"Error registering FaceNet to Laravel: {e}")
53.   return False, str(e
54.   def send_heartbeat(self, ngrok_url, facenet_model=None):
55.   """Kirim heartbeat ke Laravel server"""
56.   try:
57.   url = self.base_url + self.heartbeat_endpoint
58.   server_info = {
59.   'model_loaded': facenet_model is not None,
60.   'registered_identities':
   len(facenet_model.get_registered_identities()) if
   facenet_model else 0
61.   }
62.   payload = {
63.   'service_name': 'facenet_verification',
```

```
64.   'ngrok_url': ngrok_url,
65.   'status': 'active',
66.   'stats': API_STATS,
67.   'server_info': server_info,
68.   'timestamp': datetime.now().isoformat()
69.   }
70.   response = requests.post(
71.   url,
72.   json=payload,
73.   headers=self.get_headers(),
74.   timeout=5
75.   )
76.   if response.status_code == 200:
77.   logger.debug(f"Heartbeat sent successfully")
78.   return True
79.   else:
80.   logger.warning(f"Heartbeat failed: {response.status_code}")
81.   return False
82.   except Exception as e:
83.   logger.warning(f"Heartbeat error: {e}")
84.   return False


85.   def log_verification(self, verification_data):
86.   """Log verification ke Laravel server"""
87.   try:
88.   url = self.base_url + self.log_endpoint


89.   response = requests.post(
90.   url,
91.   json=verification_data,
92.   headers=self.get_headers(),
93.   timeout=5
94.   )
95.   if response.status_code == 200:
96.   logger.debug("Verification logged successfully")
97.   return True
98.   else:
99.   logger.warning(f"Verification log failed:
   {response.status_code}")
100.  return False
101.  except Exception as e:
102.  logger.warning(f"Verification log error: {e}")
103.  return False
```

Lampiran 4 Source Code Initialize Laravel integration

```
1. laravel_integration = LaravelIntegration(LARAVEL_CONFIG)
2. def setup_logging():
3. """Setup logging configuration"""
4. Config.create_directories()
5. logging.basicConfig(
6. level=logging.INFO,
```

```
format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
handlers=[
logging.FileHandler(os.path.join(Config.LOGS_DIR,
'facenet_api_server.log')),
  logging.StreamHandler(sys.stdout)
  ]
  )
  def create_app():
  """Create and configure Flask app"""
  app = Flask(__name__)
  CORS(app)
  # Initialize components
  face_detector = FaceDetector()
  facenet_model = FaceNetModel()
  # Load existing embeddings
  facenet_model.load_embeddings()
  logger = logging.getLogger(__name__)
  @app.route('/', methods=['GET'])
  def index():
  """API status endpoint"""
  global NGROK_URL, API_STATS
  identities = facenet_model.get_registered_identities()
  uptime_start =
datetime.fromisoformat(API_STATS['uptime_start'])
  uptime_duration = datetime.now() - uptime_start
  return jsonify({
  'status': 'running',
  'service': 'Face Verification API with FaceNet',
  'version': '1.0.0',
  'timestamp': datetime.now().isoformat(),
  'server_info': {
  'ngrok_url': NGROK_URL,
  'local_url': f"http://localhost:{Config.API_PORT}",
  'laravel_server': LARAVEL_CONFIG['base_url'],
  'uptime_hours': round(uptime_duration.total_seconds() /
3600, 2)
  },
  'model_info': {
  'registered_identities': len(identities),
  'identities': identities,
  'model_type': 'facenet'
  },
  'statistics': API_STATS
  })
  @app.route('/health', methods=['GET'])
  def health():
  """Health check endpoint"""
  return jsonify({
  'status': 'healthy',
  'timestamp': datetime.now().isoformat(),
  'model_loaded': True,
  'registered_identities':
len(facenet_model.get_registered_identities())
  })
  @app.route('/verify-face', methods=['POST'])
  def verify_face():
```

```
59.   """Face verification endpoint - Enhanced dengan Laravel
   integration"""
60.   global API_STATS
61.   start_time = time.time()
62.   request_id = str(uuid.uuid4())
63.   API_STATS['total_requests'] += 1
64.   # Initialize verification log data untuk Laravel
65.   log_data = {
66.   'request_id': request_id,
67.   'verified_identity': None,
68.   'similarity_score': None,
69.   'confidence_score': None,
70.   'face_detected': False,
71.   'verification_success': False,
72.   'processing_time_ms': None,
73.   'error_code': None,
74.   'error_message': None,
75.   'client_ip': request.remote_addr,
76.   'user_agent': request.headers.get('User-Agent', '')
77.   }
78.   # Check if image file is provided
79.   if 'image' not in request.files:
80.   API_STATS['failed_requests'] += 1
81.   log_data.update({
82.   'error_code': 'NO_IMAGE',
83.   'error_message': 'No image file provided',
84.   'processing_time_ms': round((time.time() - start_time) *
   1000, 2)
85.   })
86.   laravel_integration.log_verification(log_data)
87.   return jsonify({
88.   'success': False,
89.   'message': 'No image file provided',
90.   'code': 'NO_IMAGE',
91.   'request_id': request_id
92.   }), 400


93.   file = request.files['image']
94.   if file.filename == '':
95.   API_STATS['failed_requests'] += 1
96.   log_data.update({
97.   'error_code': 'NO_IMAGE',
98.   'error_message': 'No image file selected',
99.   'processing_time_ms': round((time.time() - start_time) *
   1000, 2)
100.  })
101.  laravel_integration.log_verification(log_data)
102.  return jsonify({
103.  'success': False,
104.  'message': 'No image file selected',
105.  'code': 'NO_IMAGE',
106.  'request_id': request_id
107.  }), 400
108.  try:
109.  # Save uploaded file temporarily
```

```
110.  temp_file = tempfile.NamedTemporaryFile(
111.  delete=False,
112.  suffix=os.path.splitext(file.filename)[1]
113.  )
114.  file.save(temp_file.name)
115.  temp_file.close()
116.  # Read image
117.  image = cv2.imread(temp_file.name)
118.  # Clean up temp file
119.  os.unlink(temp_file.name)
120.  if image is None:
121.  API_STATS['failed_requests'] += 1
122.  log_data.update({
123.  'error_code': 'INVALID_IMAGE',
124.  'error_message': 'Failed to read image file',
125.  'processing_time_ms': round((time.time() - start_time) *
   1000, 2)
126.  })
127.  laravel_integration.log_verification(log_data)
128.  return jsonify({
129.  'success': False,
130.  'message': 'Failed to read image file',
131.  'code': 'INVALID_IMAGE',
132.  'request_id': request_id
133.  }), 400
134.  # Detect face
135.  face_img, detection_confidence =
   face_detector.get_largest_face(image)
136.  if face_img is None:
137.  API_STATS['failed_requests'] += 1
138.  processing_time = (time.time() - start_time) * 1000


139.  log_data.update({
140.  'face_detected': False,
141.  'error_code': 'NO_FACE_DETECTED',
142.  'error_message': 'No face detected in the image',
143.  'processing_time_ms': round(processing_time, 2)
144.  })
145.  laravel_integration.log_verification(log_data)
146.  return jsonify({
147.  'success': False,
148.  'message': 'No face detected in the image',
149.  'code': 'NO_FACE_DETECTED',
150.  'request_id': request_id,
151.  'data': {
152.  'face_detected': False,
153.  'processing_time_ms': round(processing_time, 2)
154.  }
155.  }), 200
156.  # Face detected
157.  API_STATS['faces_detected'] += 1
158.  log_data['face_detected'] = True
159.  # Verify face dengan FaceNet
160.  verification_result = facenet_model.verify_face(face_img)
161.  # Calculate processing time
```

```
processing_time = (time.time() - start_time) * 1000
# Prepare response data
response_data = {
'face_detected': True,
'detection_confidence': round(detection_confidence, 3),
'processing_time_ms': round(processing_time, 2),
'distance': verification_result.get('distance', 0),
'similarity': verification_result.get('similarity', 0),
'confidence': verification_result.get('confidence', 0),
'request_id': request_id,
'timestamp': datetime.now().isoformat()
}
# Update log data
log_data.update({
'similarity_score': verification_result.get('similarity',
0),
'confidence_score': verification_result.get('confidence',
0),
'processing_time_ms': round(processing_time, 2)
})
if verification_result['success']:
# Successful verification
API_STATS['successful_verifications'] += 1


log_data.update({
'verified_identity': verification_result['identity'],
'verification_success': True
})
laravel_integration.log_verification(log_data)
return jsonify({
'success': True,
'message': 'Face verification successful',
'user_id': verification_result['identity'],
'name': verification_result['identity'],
'similarity': round(verification_result['similarity'], 3),
'data': response_data
}), 200
else:
# Face not recognized
API_STATS['failed_requests'] += 1
log_data.update({
'verified_identity': verification_result.get('identity'),
'verification_success': False,
'error_code': 'FACE_NOT_RECOGNIZED',
'error_message': verification_result.get('message', 'Face
not recognized')
})
laravel_integration.log_verification(log_data)
return jsonify({
'success': False,
'message': verification_result.get('message', 'Face not
recognized'),
'code': 'FACE_NOT_RECOGNIZED',
'best_match': verification_result.get('identity'),
'request_id': request_id,
```

```
'data': response_data
}), 200
except Exception as e:
API_STATS['failed_requests'] += 1
processing_time = (time.time() - start_time) * 1000
log_data.update({
'error_code': 'PROCESSING_ERROR',
'error_message': str(e),
'processing_time_ms': round(processing_time, 2)
})
laravel_integration.log_verification(log_data)
logger.error(f"Error during face verification: {e}")
return jsonify({
'success': False,
'message': f'Internal server error: {str(e)}',
'code': 'PROCESSING_ERROR',
'request_id': request_id,
'data': {
'processing_time_ms': round(processing_time, 2)
}
}), 50
@app.route('/identities', methods=['GET'])
def list_identities():
"""List all registered identities"""
try:
identities = facenet_model.get_registered_identities()
return jsonify({
'success': True,
'count': len(identities),
'identities': identities,
'timestamp': datetime.now().isoformat()
}), 200
except Exception as e:
logger.error(f"Error listing identities: {e}")
return jsonify({
'success': False,
'message': f'Error listing identities: {str(e)}'
}), 500
@app.route('/register-identity', methods=['POST'])
def register_identity():
"""Register a new identity with uploaded images"""
global API_STATS
try:
# Get identity name
if 'name' not in request.form:
return jsonify({
'success': False,
'message': 'Identity name is required'
}), 400
name = request.form['name'].strip()
if not name:
return jsonify({
'success': False,
'message': 'Identity name cannot be empty'
}), 400
# Check if images are provided
```

```
268.  if 'images' not in request.files:
269.  return jsonify({
270.  'success': False,
271.  'message': 'No image files provided'
272.  }), 400
273.  files = request.files.getlist('images')
274.  if not files or all(f.filename == '' for f in files):
275.  return jsonify({
276.  'success': False,
277.  'message': 'No valid image files provided'
278.  }), 400
279.  face_images = []
280.  processed_count = 0
281.  failed_count = 0


282.  # Process each uploaded image
283.  for file in files:
284.  if file.filename == '':
285.  continue
286.  try:
287.  # Save to temporary file
288.  temp_file = tempfile.NamedTemporaryFile(
289.  delete=False,
290.  suffix=os.path.splitext(file.filename)[1]
291.  )
292.  file.save(temp_file.name)
293.  temp_file.close()
294.  # Read and process image
295.  image = cv2.imread(temp_file.name)
296.  os.unlink(temp_file.name
297.  if image is not None:
298.  face_img, confidence = face_detector.get_largest_face(image)
299.  if face_img is not None:
300.  face_images.append(face_img)
301.  processed_count += 1
302.  else:
303.  failed_count += 1
304.  else:
305.  failed_count += 1
306.  except Exception as e:
307.  logger.error(f"Error processing uploaded image: {e}")
308.  failed_count += 1
309.  if not face_images:
310.  return jsonify({
311.  'success': False,
312.  'message': 'No valid faces found in uploaded images'
313.  }), 400
314.  # Register the identity
315.  success = facenet_model.register_face(name, face_images)
316.  if success:
317.  # Save embeddings
318.  facenet_model.save_embeddings()
319.  API_STATS['identities_registered'] += 1
320.  # Log to Laravel (optional)
321.  identity_log = {
```

```
'action': 'register_identity',
'identity_name': name,
'total_faces': len(face_images),
'processed_images': processed_count,
'failed_images': failed_count,
'timestamp': datetime.now().isoformat()
}
laravel_integration.log_verification(identity_log)


return jsonify({
'success': True,
'message': f'Identity {name} registered successfully',
'processed_images': processed_count,
'failed_images': failed_count,
'total_images': len(files),
'total_faces': len(face_images)
}), 200
else:
return jsonify({
'success': False,
'message': 'Failed to register identity'
}), 500
except Exception as e:
logger.error(f"Error registering identity: {e}")
return jsonify({
'success': False,
'message': f'Error registering identity: {str(e)}'
}), 500
@app.route('/add-faces', methods=['POST'])
def add_faces_to_identity():
"""Add more faces to existing identity - sama seperti file
asli"""
try:
# Get identity name
if 'name' not in request.form:
return jsonify({
'success': False,
'message': 'Identity name is required'
}), 400
name = request.form['name'].strip()
if not name:
return jsonify({
'success': False,
'message': 'Identity name cannot be empty'
}), 400
# Check if identity exists
existing_identities =
facenet_model.get_registered_identities()
if name not in existing_identities:
return jsonify({
'success': False,
'message': f'Identity {name} not found. Use /register-
identity for new users.'
}), 404
# Check if images are provided
```

```
373. if 'images' not in request.files:
374. return jsonify({
375. 'success': False,
376. 'message': 'No image files provided'
377. }), 400


378. files = request.files.getlist('images')
379. if not files or all(f.filename == '' for f in files):
380. return jsonify({
381. 'success': False,
382. 'message': 'No valid image files provided'
383. }), 400
384. # Get existing embedding
385. existing_embedding = facenet_model.embeddings_db[name]
386. # Process new images
387. new_face_images = []
388. processed_count = 0
389. failed_count = 0
390. for file in files:
391. if file.filename == '':
392. continue
393. try:
394. # Save to temporary file
395. temp_file = tempfile.NamedTemporaryFile(
396. delete=False,
397. suffix=os.path.splitext(file.filename)[1]
398. )
399. file.save(temp_file.name)
400. temp_file.close()
401. # Read and process image
402. image = cv2.imread(temp_file.name)
403. os.unlink(temp_file.name)
404. if image is not None:
405. face_img, confidence = face_detector.get_largest_face(image)
406. if face_img is not None:
407. new_face_images.append(face_img)
408. processed_count += 1
409. else:
410. failed_count += 1
411. else:
412. failed_count += 1
413. except Exception as e:
414. logger.error(f"Error processing uploaded image: {e}")
415. failed_count += 1
416. if not new_face_images:
417. return jsonify({
418. 'success': False,
419. 'message': 'No valid faces found in uploaded images'
420. }), 400
421. # Generate embeddings for new faces
422. new_embeddings = []
423. for face_img in new_face_images:
424. embedding = facenet_model.get_embedding(face_img)
425. if embedding is not None:
426. new_embeddings.append(embedding)
```

```
427.  if not new_embeddings:
428.  return jsonify({
429.  'success': False,
430.  'message': 'Failed to generate embeddings from new faces'
431.  }), 500
432.  # Combine with existing embedding (weighted average)
433.  existing_weight = 0.7
434.  new_weight = 0.3
435.  avg_new_embedding = np.mean(new_embeddings, axis=0)
436.  combined_embedding = (existing_weight * existing_embedding +
437.  new_weight * avg_new_embedding)
438.  # Normalize combined embedding
439.  combined_embedding = combined_embedding /
   np.linalg.norm(combined_embedding)
440.  # Update database
441.  facenet_model.embeddings_db[name] = combined_embedding
442.  # Save embeddings
443.  facenet_model.save_embeddings()
444.  return jsonify({
445.  'success': True,
446.  'message': f'Added {len(new_embeddings)} new faces to
   {name}',
447.  'processed_images': processed_count,
448.  'failed_images': failed_count,
449.  'total_images': len(files),
450.  'new_faces_added': len(new_embeddings)
451.  }), 200
452.  except Exception as e:
453.  logger.error(f"Error adding faces to identity: {e}")
454.  return jsonify({
455.  'success': False,
456.  'message': f'Error adding faces: {str(e)}'
457.  }), 500
458.  @app.route('/remove-identity', methods=['DELETE'])
459.  def remove_identity():
460.  """Remove a registered identity"""
461.  try:
462.  data = request.get_json()
463.  if not data or 'name' not in data:
464.  return jsonify({
465.  'success': False,
466.  'message': 'Identity name is required'
467.  }), 400
468.  name = data['name'].strip()
469.  if not name:
470.  return jsonify({
471.  'success': False,
472.  'message': 'Identity name cannot be empty'
473.  }), 400


474.  success = facenet_model.remove_identity(name)
475.  if success:
476.  # Save updated embeddings
```

```
facenet_model.save_embeddings()
return jsonify({
'success': True,
'message': f'Identity {name} removed successfully'
}), 200
else:
return jsonify({
'success': False,
'message': f'Identity {name} not found'
}), 404
except Exception as e:
logger.error(f"Error removing identity: {e}")
return jsonify({
'success': False,
'message': f'Error removing identity: {str(e)}'
}), 500
return app, facenet_model
```

Lampiran 5 Source Code Start ngrok dan register ke Laravel server

```
def start_ngrok_with_laravel_integration():
"""Start ngrok dan register ke Laravel server"""
global NGROK_URL
print(f"🔄 Starting ngrok tunnel for port {Config.API_PORT}...")
try:
# Check if pyngrok is installed
try:
from pyngrok import ngrok
print("✓ pyngrok module imported successfully")
except ImportError as e:
logger.error("❌ pyngrok not installed")
print("❌ pyngrok not installed. Install with: pip install pyngrok")
return
# Kill any existing ngrok processes
try:
ngrok.kill()
print("🔄 Killed existing ngrok processes")
except Exception as e:
print(f"ℹ️ No existing ngrok processes to kill: {e}")
# Start ngrok tunnel - gunakan Config.API_PORT
print(f"🚀 Creating ngrok tunnel on port {Config.API_PORT}...")
public_tunnel = ngrok.connect(Config.API_PORT)
NGROK_URL = public_tunnel.public_url
logger.info(f"🌐 Ngrok tunnel started!")
logger.info(f"🔗 Public URL: {NGROK_URL}")
print(f"✅ Ngrok tunnel successful!")
print(f"🔗 Public URL: {NGROK_URL}")
# Register to Laravel server - get facenet_model from global scope
```

```
29.   print("📡 Registering to Laravel server...")
30.   success, result = laravel_integration.register_facenet_url(
31.   NGROK_URL, f'http://localhost:{Config.API_PORT}', None
32.   )
33.   if success:
34.   logger.info("✓ Successfully registered to Laravel server")
35.   print("✅ Successfully registered to Laravel server")
36.   else:
37.   logger.warning(f"⚠ Failed to register to Laravel:
   {result}")
38.   print(f"⚠ Failed to register to Laravel: {result}")
39.   print_api_examples()
40.   except ImportError as e:
41.   logger.error(f"❌ Import error: {e}")
42.   print(f"❌ pyngrok import error: {e}")
43.   print("💡 Try installing pyngrok:")
44.   print("pip install pyngrok")
45.   except Exception as e:
46.   logger.error(f"❌ Failed to start ngrok: {e}")
47.   print(f"❌ Ngrok failed to start: {e}")
48.   print("💡 Troubleshooting steps:")
49.   print("1. Check if ngrok is installed: ngrok --version")
50.   print("2. Try setting ngrok auth token:")
51.   print("ngrok config add-authtoken YOUR_TOKEN")
52.   print("3. Check if port is available:")
53.   print(f"netstat -an | grep {Config.API_PORT}")
```