MOSIP OCR Field Extraction & Verification System - Documentation

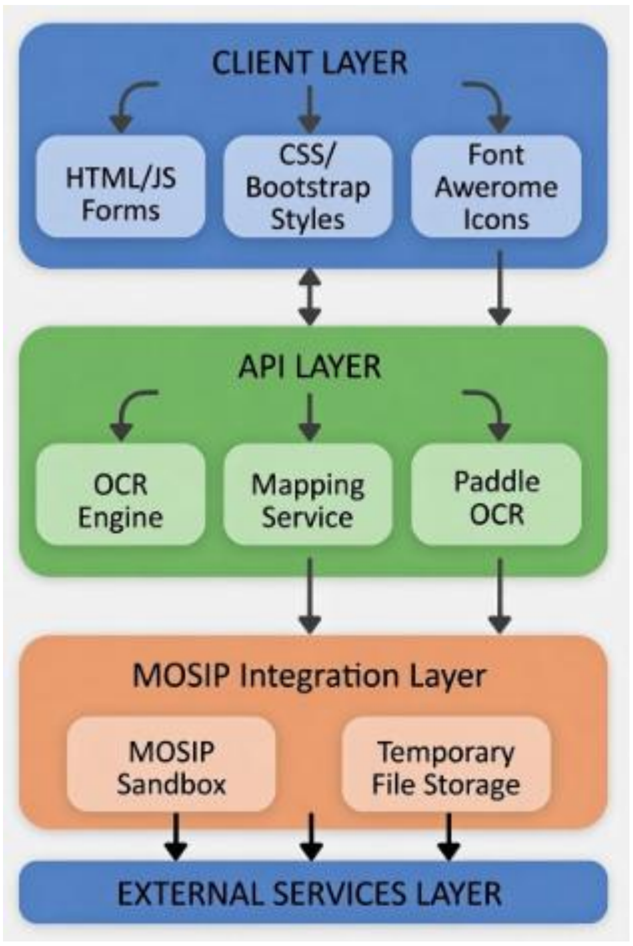
1. Detailed Workflow Documentation

1.1 Architectural Design

System Overview

The MOSIP OCR Field Extraction & Verification System is a **three-tier web application** that automates document processing for MOSIP (Modular Open Source Identity Platform) pre-registration. The system extracts text from identity documents, verifies extracted data against user input, and integrates with MOSIP's pre-registration API.

Architecture Diagram



Technology Stack

Component	Technology	Purpose
Frontend	HTML5, JavaScript, Bootstrap 5	User interface for document upload and data review
Backend	Python FastAPI, Uvicorn	REST API server for OCR processing
OCR Engine	PaddleOCR	Text extraction from images/PDFs with multi-language support
Image Processing	OpenCV, Pillow	Image preprocessing for better OCR accuracy
MOSIP Integration	Requests library	Communication with MOSIP pre-registration API
Development Server	Python http.server	Frontend hosting during development
Data Storage	In-memory/File system	Temporary storage of uploaded documents

Design Patterns

1. **RESTful API Design** - Stateless, resource-oriented endpoints
2. **Middleware Pattern** - CORS handling, request/response processing
3. **Adapter Pattern** - MOSIP API client with mock/real adapters
4. **Factory Pattern** - OCR processor creation based on file type
5. **Strategy Pattern** - Different verification strategies per document type

Security Considerations

- **No persistent storage** of sensitive documents
- **In-memory processing** with automatic cleanup
- **CORS restricted** to frontend origins
- **Environment-based configuration** for sensitive data
- **Input validation** on all API endpoints

- **File type validation** and size limits

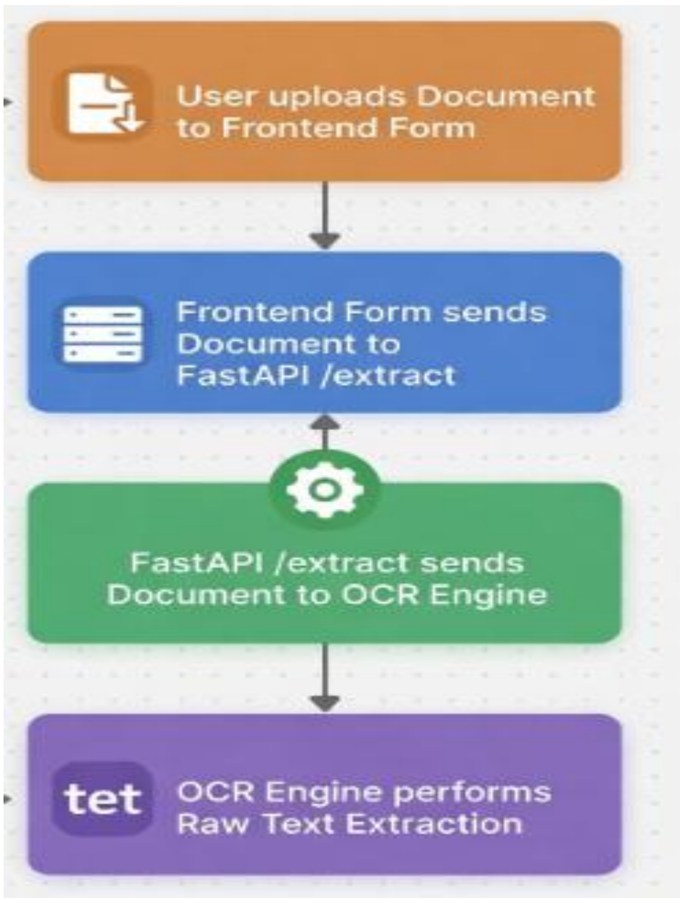
1.2 Data Flow Structure

Primary Workflow: Document Processing

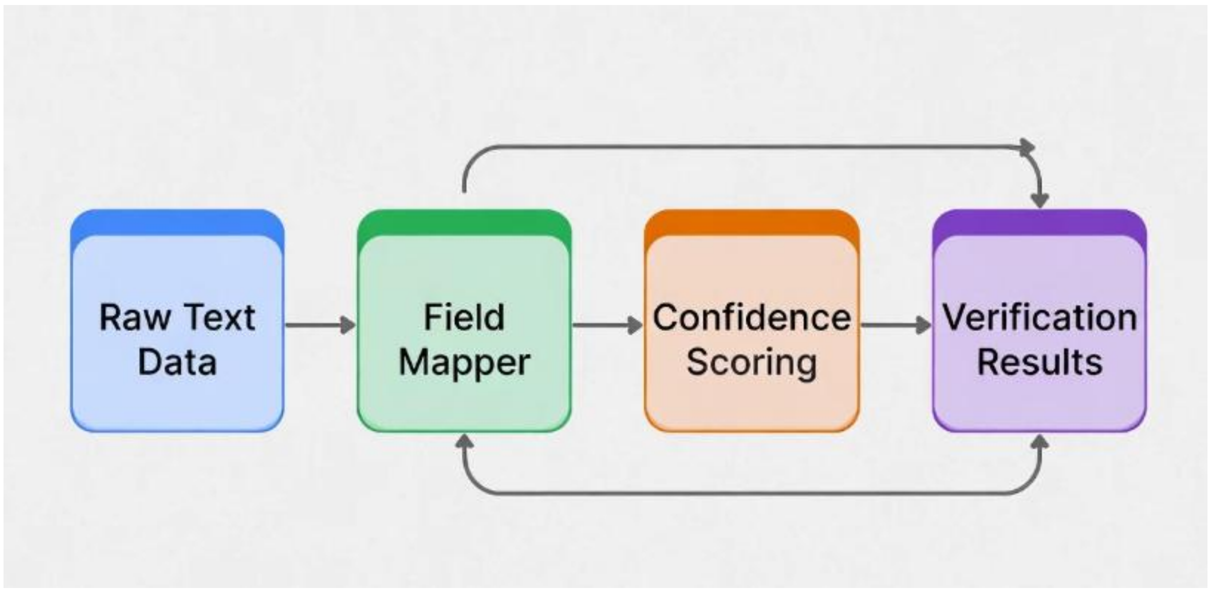
1. User Upload → 2. OCR Extraction → 3. Data Mapping → 4. Verification → 5. MOSIP Submission

Detailed Data Flow

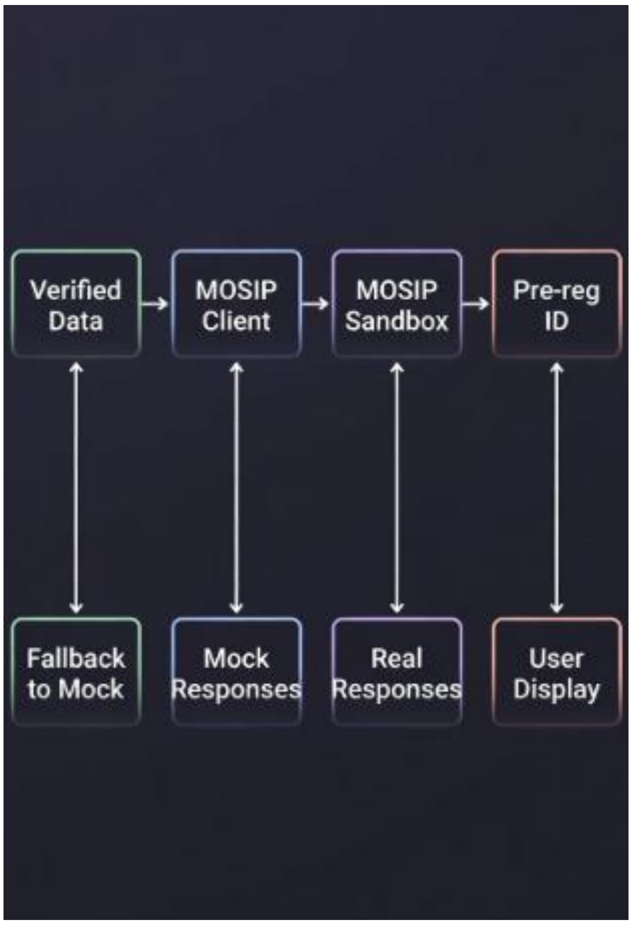
Phase 1: Document Upload & OCR Extraction


User uploads Document to Frontend Form
Frontend Form sends Document to FastAPI /extract
FastAPI /extract sends Document to OCR Engine
tet
OCR Engine performs Raw Text Extraction

Phase 2: Field Mapping & Verification



Phase 3: MOSIP Integration



Data Transformation Pipeline

Input Formats:

- Images: JPG, PNG, TIFF
- Documents: PDF
- Maximum size: 16MB per file

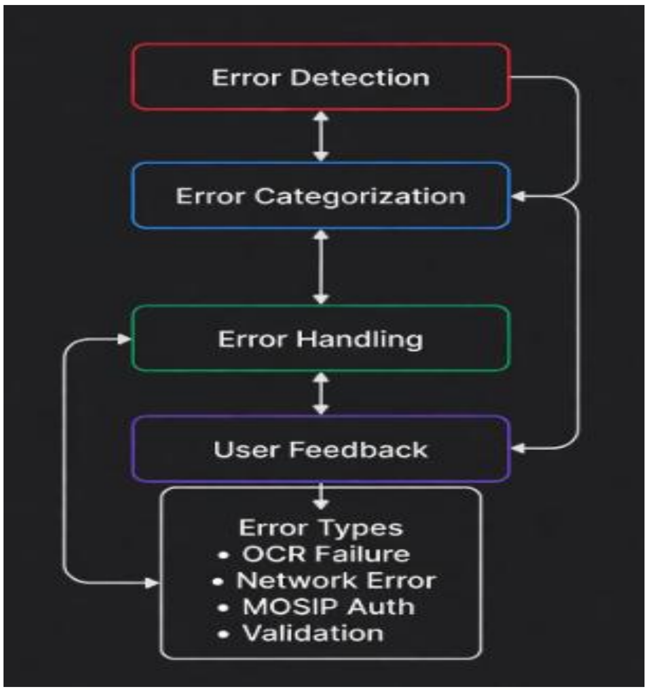
Processing Steps:

1. **Preprocessing**: Image enhancement, rotation correction, noise reduction
2. **OCR Processing**: Language-specific text extraction (English primary)
3. **Field Identification**: Regex patterns and keyword matching
4. **Confidence Scoring**: Per-field accuracy assessment
5. **Data Validation**: Format verification (dates, emails, phone numbers)
6. **MOSIP Schema Mapping**: Conversion to MOSIP-compatible format

Output Formats:

- JSON structured data
- Confidence scores (0.0-1.0)
- Verification status
- MOSIP pre-registration ID (or mock ID)

Error Handling Flow



1.3 Integration and Installation Guidance

Prerequisites:

```
# System Requirements
- Python 3.8 or higher
- **No system-wide OCR installation needed** (PaddleOCR includes models)
- 4GB RAM minimum (more for multi-language)
- 2GB free disk space for PaddleOCR models
```

Installation Steps:

Step 1: Clone and Setup

```
# Clone the repository
git clone <repository-url>
cd ocr-mosip-integration

# Create virtual environment (Windows)
python -m venv venv
venv\Scripts\activate

# Or on Mac/Linux
python3 -m venv venv
source venv/bin/activate
```

Step 2: Install Python Dependencies with PaddleOCR

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

# Updated requirements.txt contents:
fastapi
uvicorn[standard]
paddlepaddle
paddleocr
pillow
opencv-python
pdf2image
requests
python-dotenv
```

Step 3: Install Python Dependencies

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

# Key dependencies:
# fastapi, uvicorn, pytesseract, pillow, opencv-python,
# pdf2image, requests, python-dotenv
```

Step 4: Configure Environment

```
# Create .env file from template
cp .env.example .env

# Edit .env with your configuration
# MOSIP_BASE_URL=https://sandbox.mosip.net
# MOSIP_AUTH_TOKEN=your_token_here
# DEBUG=True
```

Step 5: Run the Application

```
# Terminal 1: Backend API
cd backend
python -m uvicorn app:app --reload --port 8000

# Terminal 2: Frontend Server
cd frontend
python -m http.server 5000
```

Docker Deployment (Alternative)

```
# Dockerfile
FROM python:3.9-slim
RUN apt-get update && apt-get install -y tesseract-ocr
WORKDIR /app
COPY backend/requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "8000"]
```

Configuration Management

Environment Variables:

```
# OCR Configuration
OCR_ENGINE=paddleocr
OCR_LANGUAGES=en,ar,hi  # Multiple languages supported
OCR_USE_GPU=False
```

```
# Required
MOSIP_BASE_URL=https://sandbox.mosip.net
MOSIP_AUTH_TOKEN=your_authentication_token

# Optional with defaults
DEBUG=True
PORT=8000
VERIFICATION_THRESHOLD=0.85
MAX_UPLOAD_SIZE=16777216
ALLOWED_EXTENSIONS=.pdf,.png,.jpg,.jpeg
LOG_LEVEL=INFO
```

File Structure:

```
ocr-mosip-integration/
├── backend/
│   ├── app.py                      # FastAPI application
│   ├── requirements.txt            # Python dependencies
│   ├── core/
│   │   ├── ocr.py                 # OCR processing engine
│   │   ├── mapper.py              # Field mapping logic
│   │   ├── verifier.py            # Verification engine
│   │   └── mosip_client.py        # MOSIP integration
│   ├── routes/
│   │   ├── extraction.py          # OCR endpoints
│   │   ├── mapping.py             # Field mapping endpoints
│   │   ├── verification.py        # Verification endpoints
│   │   └── mosip.py               # MOSIP endpoints
│   └── tests/                     # Test files
├── frontend/
│   ├── index.html                 # Main application
│   ├── app.js                     # Frontend logic
│   ├── styles.css                 # Styling
│   └── mosip.html                # MOSIP integration UI
├── .env                           # Environment configuration
├── .env.example                  # Configuration template
├── .gitignore                    # Git ignore rules
└── README.md                     # Project documentation
```

Integration Points

With MOSIP Sandbox:

- **Base URL**: https://sandbox.mosip.net
- **Authentication**: Bearer token in Authorization header
- **Endpoints Used**:
  - /preregistration/v1/applications (POST) - Create application
  - /preregistration/v1/documents/{id} (POST) - Upload documents
  - /preregistration/v1/applications/{id} (GET) - Check status

With External OCR Services (Future):

- Google Vision API
- AWS Textract

- Azure Computer Vision

Monitoring & Logging:

```
# Logging configuration
logging.basicConfig(
    level=os.getenv('LOG_LEVEL', 'INFO'),
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
)
```

Troubleshooting Guide

Common Issues:

1. **OCR not working**: Install Tesseract and language packs
2. **MOSIP connection failed**: Check token and network connectivity
3. **File upload errors**: Verify file size and format restrictions
4. **CORS errors**: Ensure frontend origin is in allowed list

Debug Mode:

```bash
# Set debug mode in .env
DEBUG=True
LOG_LEVEL=DEBUG

# Check logs for detailed information
```

2. API Documentation

2.1 API Overview

The MOSIP OCR Field Extraction & Verification System provides RESTful APIs for document processing, verification, and MOSIP integration.

Base URL: http://localhost:8000 (development)
Content-Type: application/json for JSON endpoints, multipart/form-data for file uploads
API Version: v1

2.2 Authentication

Currently, the API uses **no authentication** for OCR endpoints. MOSIP endpoints require authentication via:

```http
Authorization: Bearer {MOSIP_AUTH_TOKEN}
```

Request Body:

```form-data
file: (required) Document file (PDF, JPG, PNG, TIFF)
```

Response (Success - 200):

```
{
  "status": "success",
  "extracted_text": "Full extracted text here...",
  "file_name": "document.pdf",
  "file_type": "application/pdf",
  "processing_time": 2.34,
  "engine": "paddleocr",
  "confidence": 0.92
}
```

Response (Error - 400):

```
{
  "status": "error",
  "message": "No file uploaded",
  "code": "NO_FILE"
}
```

cURL Example:

```
curl -X POST http://localhost:8000/api/v1/ocr/extract-text \
  -F "file=@id_card.jpg"
```

A2. Health Check

Endpoint: GET /api/v1/ocr/health
Description: Checks OCR service health and configuration

Response:

```
{
  "status": "healthy",
  "service": "ocr-extraction",
  "version": "1.0.0",
  "engine": "paddleocr",
  "supported_languages": ["en", "ar", "hi"],
  "timestamp": "2024-01-15T10:30:00Z"
}
```

Category B: Field Mapping & Verification APIs

B1. Map and Verify Fields

Endpoint: POST /api/v1/map-and-verify
Description: Maps extracted text to structured fields and verifies against user input
Content-Type: application/json

Request Body:

```
{
  "raw_text": "Name: John Doe\nAge: 30\nAddress: 123 Main Street",
  "user": {
    "name": "John Doe",
    "dob": "1993-01-15",
    "gender": "Male",
    "address": "123 Main Street",
    "email": "john@example.com",
    "phone": "+911234567890"
  },
  "document_type": "aadhar"  // Optional: "aadhar", "passport", "dl", "birth"
}
```

B2. Batch Verification

Endpoint: POST /api/v1/batch-verify
Description: Verify multiple documents at once

Request Body:

```
{
  "documents": [
    {
      "raw_text": "Name text...",
      "user_data": {"name": "..."},
      "type": "aadhar"
    }
  ]
}
```

Category C: MOSIP Integration APIs

C1. Complete MOSIP Integration

Endpoint: POST /api/v1/mosip/integrate
Description: Complete workflow: OCR → Verification → MOSIP Registration
Content-Type: multipart/form-data

Request Body:

```form-data
file: (required) Document file
manual_data: (optional) JSON string for verification
verification_threshold: (optional) Confidence threshold (default: 0.85)
```

Response (Success - 200):

```
{
  "status": "success",
  "message": "Successfully registered with MOSIP",
  "pre_registration_id": "PRE123456789",
  "extracted_data": {
    "Name": "John Doe",
    "Gender": "Male"
  },
  "verification_results": {
    "Name": 0.98
  },
  "mode": "real",  // or "mock" if using demo token
  "next_steps": {
    "check_status": "/api/v1/mosip/status/PRE123456789",
    "mosip_portal": "https://sandbox.mosip.net/pre-registration"
  }
}
```

C2. Verify and Submit

Endpoint: POST /api/v1/mosip/verify-and-submit
Description: Verify already extracted data and submit to MOSIP
Content-Type: application/json

```
{
  "extracted_data": {
    "Name": "John Doe",
    "DOB": "1993-01-15"
  },
  "manual_data": {
    "Name": "John Doe"
  },
  "skip_verification": false
}
```

C3. Check Registration Status

Endpoint: GET /api/v1/mosip/status/{pre_reg_id}
Description: Check status of MOSIP pre-registration

Response:

```json
{
  "pre_registration_id": "PRE123456789",
  "status": {
    "application_status": "PENDING",
    "documents_status": "UPLOADED",
    "last_updated": "2024-01-15T10:30:00Z"
  }
}
```

C4. Test MOSIP Connection

Endpoint: GET /api/v1/mosip/test
Description: Test connectivity to MOSIP sandbox

Response:

```json
{
  "status": "connected",
  "mosip_base_url": "https://sandbox.mosip.net",
  "authenticated": true,
  "mode": "real"
}
```

C5. Batch Submit

Endpoint: POST /api/v1/mosip/batch-submit
Description: Submit multiple documents to MOSIP

Request Body: multipart/form-data

```form-data
files: (required) Multiple files
verification_data: (optional) JSON array of verification data
```

2.4 Common Error Responses

HTTP Status Codes

- 200 OK: Request successful
- 400 Bad Request: Invalid input or missing parameters
- 401 Unauthorized: MOSIP authentication failed
- 404 Not Found: Endpoint or resource not found
- 415 Unsupported Media Type: Invalid file format

- 422 Unprocessable Entity: Validation error
- 500 Internal Server Error: Server-side error
- 502 Bad Gateway: MOSIP service unavailable

Error Response Format

```
{
  "status": "error",
  "message": "Human readable error message",
  "code": "ERROR_CODE",
  "details": {
    "field": "Specific error details"
  },
  "timestamp": "2024-01-15T10:30:00Z"
}
```

Common Error Codes:

Code	Description	Resolution
NO_FILE	No file uploaded	Include a file in the request
INVALID_FILE_TYPE	Unsupported file format	Use JPG, PNG, PDF, or TIFF
FILE_TOO_LARGE	File exceeds size limit	Upload files < 16MB
OCR_FAILED	Text extraction failed	Check image quality
MOSIP_AUTH_FAILED	MOSIP authentication failed	Check MOSIP_AUTH_TOKEN
MOSIP_UNAVAILABLE	MOSIP service down	Try again later
VALIDATION_ERROR	Input validation failed	Check request format

2.5 Rate Limiting

- **OCR Endpoints**: 10 requests per minute per IP
- **MOSIP Endpoints**: 5 requests per minute per IP

- **Batch Operations**: 2 requests per minute per IP

2.6 Request/Response Examples

Example 1: Complete OCR to MOSIP Flow

```
# Step 1: Extract text
curl -X POST http://localhost:8000/api/v1/ocr/extract-text \
  -F "file=@aadhar_card.jpg"

# Step 2: Map and verify
curl -X POST http://localhost:8000/api/v1/map-and-verify \
  -H "Content-Type: application/json" \
  -d '{
    "raw_text": "...extracted text...",
    "user": {"name": "John Doe"}
  }'

# Step 3: Submit to MOSIP
curl -X POST http://localhost:8000/api/v1/mosip/integrate \
  -F "file=@aadhar_card.jpg" \
  -F "manual_data={\"Name\": \"John Doe\"}"
```

Example 2: Direct MOSIP Submission with Verification

```
import requests

# Prepare document
files = {'file': open('document.pdf', 'rb')}
data = {'manual_data': '{"Name": "John Doe"}'}

# Submit to MOSIP
response = requests.post(
    'http://localhost:8000/api/v1/mosip/integrate',
    files=files,
    data=data
)

print(f"Pre-registration ID: {response.json()['pre_registration_id']}")
```

2.7 API Testing

Using OpenAPI/Swagger

Visit: http://localhost:8000/docs for interactive API testing

Test Script

```python
import requests
import json

BASE_URL = "http://localhost:8000"

def test_all_endpoints():
    endpoints = [
        ("GET", "/", {}),
        ("GET", "/api/v1/ocr/health", {}),
        ("POST", "/api/v1/mosip/test", {})
    ]

    for method, endpoint, data in endpoints:
        url = BASE_URL + endpoint
        try:
            if method == "GET":
                r = requests.get(url)
            elif method == "POST":
                r = requests.post(url, json=data)

            print(f"{method} {endpoint}: {r.status_code}")
            if r.status_code != 200:
                print(f"  Error: {r.text}")
        except Exception as e:
            print(f"{method} {endpoint}: ERROR - {e}")

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

2.8 WebSocket Endpoints (Future)

- ws://localhost:8000/ws/ocr-progress: Real-time OCR progress updates
- ws://localhost:8000/ws/verification: Live verification results

2.9 Data Models

OCR Response Model

```typescript
interface OCRResponse {
  status: 'success' | 'error';
  extracted_text: string;
  file_name: string;
  file_type: string;
  processing_time: number;
  engine: 'paddleocr';
  confidence: number;
  error?: string;
}
```

Verification Response Model:

```
interface VerificationResponse {
  status: 'success' | 'error';
  mapped_fields: {
    [field: string]: {
      value: string;
      confidence: number;
      verified: boolean;
      match_score: number;
      note?: string;
    }
  };
  verification: {
    overall_confidence: number;
    verified_fields: string[];
    warnings: string[];
    score: number;
  };
}
```

MOSIP Response Model:

```
interface MOSIPResponse {
  status: 'success' | 'error';
  message: string;
  pre_registration_id?: string;
  extracted_data: Record<string, any>;
  verification_results: Record<string, number>;
  mode: 'real' | 'mock';
  next_steps: {
    check_status: string;
    mosip_portal: string;
  };
}
```

3. Test Cases, Scenarios & Example API Responses

3.1 Core Mapper Scenarios (Unit Tests)

• Label:value fallback — Expect parsed name, address, phone.

• Regex (English + Hindi) — Expect non-empty extracted name.

• Noisy labels / multiple colons — Full address retained.

• Document-type filter — Aadhaar returns only name.

• Phone normalization — Last 10 digits extracted.

• Pincode noise — Digits extracted (define expected behavior).

• Empty / no match — Empty dict returned, no crash.

3.2 Core Verifier Scenarios (Unit Tests)

• Unicode/transliteration name match — score ≥ 0.85.

• DOB parsing exact — score = 1.0.

• Phone NSN match — score ≈ 1.0.

• Address fuzzy match — score > 0.8.

• Weighted aggregation — Uses predefined weights.

• Decision thresholds — MATCH / REVIEW / MISMATCH.

• Partial overlap — Verification skips gracefully.

• Incomplete sets — Works even when some fields missing.

3.3 API Scenarios (Integration Tests)

• Health endpoint — 200 healthy response.

• Extract-text missing file — 400 NO_FILE.

• Extract-text mocked OCR — Proper JSON output.

• Map-and-verify — Successful mapping with scores.

• MOSIP test — 200 connected.

• MOSIP integrate happy path — Returns pre_registration_id.

• MOSIP integrate failure — 400 verification_failed.

• Batch submit — Structured per-file output.

3.4 Error and Edge Scenarios

• Invalid file type — 415 INVALID_FILE_TYPE.

• File too large — 413 or FILE_TOO_LARGE.

• No overlapping fields — Verification skipped.

• OCR failure — 500 OCR_FAILED.

• MOSIP unavailable — 502 MOSIP integration error.

3.5 End-to-End (E2E) Scenarios

• Happy flow — Extract → Verify MATCH → Submit to MOSIP.

• Edit flow — User corrects OCR text and confidence improves.

• Batch mixed docs — Aadhaar, passport, handwritten handled correctly.

3.6 Security Scenarios

• Malicious text input — Returns safe 4xx, no crash.

• Path traversal filenames — Sanitized correctly.

• CORS blocking — Unauthorized origins blocked.

• Temporary file cleanup — Ensured when applicable.

3.7 Performance / Rate / Reliability Scenarios

• Large PDF — Completes under time limit.

• Rate limit — Exceeding limit returns 429.

• MOSIP retry — Graceful fallback when unreachable.

3.8 Test Data Snippets

• SAMPLE_AADHAR_TEXT example.

• SAMPLE_PASSPORT_TEXT example.

• fake_pdf and fake_image fixtures.

• Unicode + Hindi + malformed DOB samples.

3.9 CI Smoke Tests

• GitHub Actions runs pytest with mocked OCR.

• Optional flake8 linting.

3.10 Example API Request/Response Set

Mapper Function (Direct Example)

Input:

"Name: John Doe\nAddress: 123 A St\nPhone number: +91-9876543210"

Output:

{"name": "John Doe", "address": "123 A St", "phone": "9876543210"}

Health Endpoint (200)

{"status":"healthy","engine":"paddleocr","languages":["en"],"version":"1.0.0"}

Extract Text (200)

{"file_name":"sample.png","extracted_text":"Name: John Doe\nDOB: 1990-01-01"}

Extract Text Missing File (400)

{"detail":"NO_FILE"}

Map and Verify (200)

{"status":"success","mapped":{"name":"John Doe","dob":"1990-01-01"},"missing_fields":[],"verification":{"overall_confidence":0.95,"decision":"MATCH","fields":{"name":0.95, "dob":1.0},"notes":[]}}

MOSIP Test (200)

{"status":"connected","mosip_base_url":"https://sandbox.mosip.net","test_response":{"status":"success"}}

MOSIP Integrate (200)

{"status":"success","message":"Successfully registered with MOSIP","pre_registration_id":"PREABC12345"}

Verification Failure (400)

{"status":"verification_failed","threshold":0.8}

Batch Submit (200)

{"batch_id":"batch_ab12cd34","total_files":2}