
Patient Assessment Form Processor Documentation

---

1. Overview

I developed the **Patient Assessment Form Processor** to automate the extraction of data from patient assessment forms using Optical Character Recognition (OCR) and to store the results in a PostgreSQL database. This project is designed to accurately extract both handwritten and printed text from image files (JPEG or PDF), structure the data into JSON format, and then integrate it into a SQL database. In this document, I explain my approach, the technologies I used, and the steps I took to complete the assignment.

---

2. Intern Assignment Overview

For this assignment, I was responsible for:

- **Developing a Python-Based OCR Script:**
    - I implemented a Python script that extracts text from patient assessment forms.
    - I ensured the script can handle both handwritten and printed text.
    - I converted the extracted text into a structured JSON format.
- **Storing Extracted Data in a SQL Database:**
    - I designed a robust database schema to store patient data and form results.
    - I inserted the structured JSON data into the relevant database tables in PostgreSQL.
- **Providing a Public GitHub Repository:**
    - I created a public GitHub repository with all source code, documentation, and setup instructions.
    - I made sure the repository was accessible and submitted the repository link before the interview.

---

3. Detailed Step-by-Step Instructions

3.1 Data Extraction Using OCR

a. Tools and Libraries

- **OCR Engine:**
  I used Tesseract OCR (though EasyOCR was also considered) for extracting text from images.

- **Image Processing Libraries:**
  I employed OpenCV and Pillow to preprocess images and enhance OCR accuracy.

b. Image Preprocessing

To improve OCR accuracy, I performed the following steps:

- **Convert to Grayscale:**
  I converted images to grayscale to simplify the data and improve text clarity.
- **Thresholding:**
  I applied binary thresholding to distinguish text from the background.
- **Noise Removal:**
  I used filtering techniques (e.g., blurring) to reduce background interference.

c. Extracting Text

After preprocessing, I applied OCR to extract text from the forms. I verified the accuracy by testing the script on several sample forms provided in the dataset.

3.2 Identify and Extract Key Data Points

My OCR script extracts the following fields:

- **Patient Details:**
  - Patient Name
  - Date of Birth (DOB)
- **Treatment Details:**
  - Assessment Date
  - Injection (Yes/No)
  - Exercise Therapy (Yes/No)
- **Difficulty Ratings (Scale 0-5):**
  - Bending or Stooping

    - Putting on Shoes
    - Sleeping
    - Standing for an Hour
    - Going Up/Down a Flight of Stairs
    - Walking Through a Store
    - Driving for an Hour
    - Preparing a Meal
    - Yard Work
    - Picking Up Items off the Floor
- **Patient Changes:**
    - Since Last Treatment
    - Since the Start of Treatment
    - Last 3 Days (Good/Bad)
- **Pain Symptoms (Scale 0-10):**
    - Pain
    - Numbness
    - Tingling
    - Burning
    - Tightness
- **Medical Assistant (MA) Inputs:**
    - Blood Pressure
    - Heart Rate (HR)
    - Weight
    - Height
    - SpO_2 (Oxygen Saturation)
    - Temperature
    - Blood Glucose
    - Respirations

I utilized regular expressions (regex) and string parsing techniques to identify these data points. I also implemented helper functions (located in `validators.py`) to validate and format the extracted data.

3.3 Convert Extracted Data to JSON

a. JSON Format Example

I structured the extracted data into JSON. Below is an example of the output:

{

```
  "patient_name": "John Doe",
  "dob": "01/05/1988",
  "date": "02/06/2025",
  "injection": "Yes",
  "exercise_therapy": "No",
  "difficulty_ratings": {
    "bending": 3,
    "putting_on_shoes": 1,
    "sleeping": 2
  },
  "patient_changes": {
    "since_last_treatment": "Not Good",
    "since_start_of_treatment": "Worse",
    "last_3_days": "Bad"
  },
  "pain_symptoms": {
    "pain": 2,
    "numbness": 5,
    "tingling": 6,
    "burning": 7,
    "tightness": 5
  },
  "medical_assistant_data": {
    "blood_pressure": "120/80",
    "hr": 80,
    "weight": 67,
    "height": "5'7",
    "spo2": 98,
    "temperature": "98.6",
    "blood_glucose": 115,
    "respirations": 16
  }
}
```


b. Implementation Details

I created a Python script named `ocr_processing.py` that:

- Reads an input image file (JPEG/PDF).
- Applies image preprocessing to enhance OCR results.
- Uses the OCR engine to extract and process the text.
- Parses the text to build the JSON structure.
- Saves the output JSON file in the `data/json_output/` directory.

3.4 Store JSON Data in SQL Database

a. Database Schema

I designed the following tables for PostgreSQL:

Patients Table:

```
 CREATE TABLE patients (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255),
  dob DATE
);
```

-

Forms Data Table:

```
 CREATE TABLE forms_data (
  id SERIAL PRIMARY KEY,
  patient_id INT REFERENCES patients(id),
  form_json JSONB,
  created_at TIMESTAMP DEFAULT NOW()
);
```

-

b. Insertion Process

I developed a separate Python script named `database.py` to handle database operations. This script:

- Reads the JSON output file.
- Inserts patient details into the `patients` table.
- Inserts the full JSON data into the `forms_data` table, linking the records using the patient ID.

---

4. Repository and Submission Guidelines

a. GitHub Repository

I created a public GitHub repository for the project at:
https://github.com/syedsami1/patient-assessment-ocr.git

The repository includes:

- All source code (organized in the `src/` and `tests/` directories).
- SQL schema files (located in `sql_schemas/schema.sql`).
- Sample JSON output (found in `data/json_output/output.json`).
- Comprehensive documentation (this document and additional guides).

b. Submission Requirements

For my submission, I ensured that the repository contains:

- A fully working OCR script.
- Structured JSON output as demonstrated.
- The database schema and data storage implementation.
- All necessary documentation and setup instructions.

---

5. Directory Structure

Below is the project's directory structure:

```
patient-assessment-ocr/
├── .env.example              // Environment variable template
├── .gitignore                // Specifies files to ignore in Git
├── README.md                 // Project documentation & setup guide
├── requirements.txt          // Python dependencies
├── src/
│   ├── ocr_processing.py     // Main OCR processing script
│   ├── database.py           // Database interaction logic
│   └── helpers/              // Utility modules
│       ├── image_utils.py    // Image preprocessing functions
│       └── validators.py     // Data validation functions
├── data/
│   ├── sample_forms/         // Sample input documents
│   │   ├── sample_form.jpg   // Example form (provided by user)
│   │   └── blank_form.pdf    // Blank template (provided by user)
│   └── json_output/          // Generated JSON files
│       └── output.json       // Example JSON output
├── sql_schemas/
│   └── schema.sql            // Database schema definition
├── tests/                    // Unit tests
```

```
│   ├── test_ocr_processing.py
│   └── test_database.py
├── config/                 // Configuration files
│   └── patterns.yaml           // Regex patterns for data extraction
└── docs/                   // Additional documentation
    └── workflow.md             // System architecture diagram/explanation
```

---

6. Setup and Execution

6.1 Clone the Repository

I cloned the repository using:

```
git clone https://github.com/syedsami1/patient-assessment-ocr.git
cd patient-assessment-ocr
```

6.2 Create a Virtual Environment and Install Dependencies

I set up a virtual environment and installed all dependencies:

```
python -m venv venv
source venv/bin/activate   # For Linux/Mac
# For Windows, use: venv\Scripts\activate
pip install -r requirements.txt
```

6.3 Configure Environment Variables

I copied the example environment file and updated the credentials:

```
cp .env.example .env
```

6.4 Set Up the Database

I created the PostgreSQL database and initialized it with the schema:

```
createdb medical_db
psql medical_db -f sql_schemas/schema.sql
```

6.5 Run the OCR Script

I processed a sample form using:

python src/ocr_processing.py -i data/sample_forms/sample_form.jpg

6.6 Store the JSON Data in the Database

I inserted the JSON output into the database with:

python src/database.py --store data/json_output/sample_output.json

6.7 Run Tests

Finally, I ran the test suite to verify functionality:

pytest tests/

---

7. Additional Notes

- **Error Handling:**
  I incorporated comprehensive error handling throughout the scripts to manage missing or improperly formatted data.

- **Data Validation:**
  I used helper functions (in `validators.py`) to validate data formats and ensure numeric values and dates adhere to expected ranges.

- **Documentation Updates:**
  I maintained the documentation in parallel with the code. Additional guides and a system architecture diagram are available in the `docs/` folder.

---

8. Final Submission

For my final submission, I pushed the complete project to my public GitHub repository. I verified that all components are present:

- The working OCR script.
- Sample JSON output.
- SQL schema and database integration scripts.
- Comprehensive documentation and setup instructions.

Repository Link:
https://github.com/syedsami1/patient-assessment-ocr.git