Screen2json

Turn any screenshot into structured, usable JSON.
Screenshots are visual. Software needs structure.
Screen2JSON bridges the gap.

Overview

This is a Receipt Scanner app that uses OCR - (Optical Character Recognition) and AI to extract structured data from receipt images. Upload a photo of a receipt, and it automatically identifies the merchant, items, prices, and totals - returning everything as a clean, structured JSON

Problem & Impact

The problem

- UI designs
- Dashboards
- Receipts
- Tables and forms

Why This Matters

- Time wasted retyping data
- Human errors
- Slower workflows for teams

Who This Helps

- Developers
- Product managers
- Operations & finance teams
- Students and hackathon teams

The Solution

What Screen2JSON Does

1. Upload a screenshot
2. AI analyzes the visual content
3. Outputs clean, structured JSON

Result

- Visual data → machine-readable data
- Ready for code, APIs, and automation

Technology Stack

StreamLit: Web Framework for the UI

Pytesseract Python Wrapper for Tesseract OCR

Opencv-python: Image preprocessing (densoising, contrast)

Pillow - Image loading and manipulation

Openai - API client for GPT models

Pydantic - Data validation and schema enforcement

Pipeline

Open the PNG
Image processing
Grey scale conversion
Upscale the Image
Extract Text Via Tesseract
Tesseract runs 6 different configurations
3 PSM - Page Segmentation Mode
Tells Tesseract how to interpret the layout of the text in the image
X2 image variants
Send OCR text to OpenAI GPT 4.1 Mini
GPT converts the text to JSON
Validate JSON output with Pydantic
Display with Streamlit

OCR (Optical Character Recognition)

- Conversion of images of typed, handwritten, or printed text to machine encoded text
- Digitizes printed texts for electronic editing, searching, storing, displaying.
- Field of Pattern Recognition, Artificial Intelligence, Machine Learning, Computer Vision

Why It’s Powerful

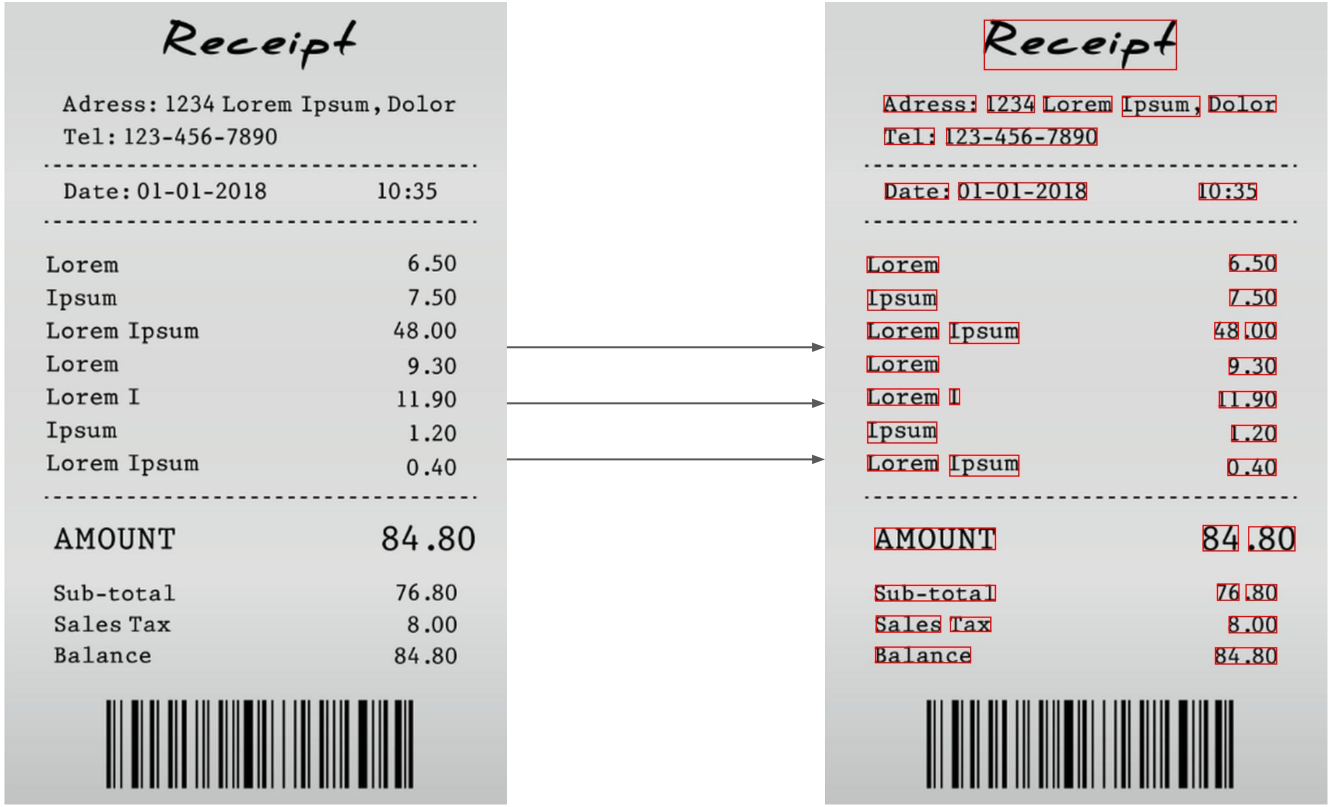
Why Screen2JSON matters

- Not just text extraction
- Preserves structure and hierarchy
- Works on messy, real screenshots

Example Use Cases

- UI screenshot → backend mock data
- Receipt → accounting system
- Dashboard → structured analytics

How Tesseract Works



Receipt

Adress: 1234 Lorem Ipsum, Dolor
Tel: 123-456-7890

Date: 01-01-2018	10:35
Lorem	6.50
Ipsum	7.50
Lorem Ipsum	48.00
Lorem	9.30
Lorem I	11.90
Ipsum	1.20
Lorem Ipsum	0.40
AMOUNT	84.80
Sub-total	76.80
Sales Tax	8.00
Balance	84.80

Receipt

Adress: 1234 Lorem Ipsum, Dolor
Tel: 123-456-7890

Date: 01-01-2018 10:35
Lorem 6.50
Ipsum 7.50
Lorem Ipsum 48.00
Lorem 9.30
Lorem I 11.90
Ipsum 1.20
Lorem Ipsum 0.40
AMOUNT 84.80
Sub-total 76.80
Sales Tax 8.00
Balance 84.80

Tesseract -> AI Pipeline

We preprocess the image to increase contrast for easier computer vision

Parses the image data into strings of text

We run multiple passes on the image to maximize correctness of extracted text, using scoring heuristics

ML Rewards: Receipt Keywords: ‘total’, ‘tax’, ‘subtotal’

Item-price patterns: ‘Milk $3.50

Money Amounts: ‘$12.34

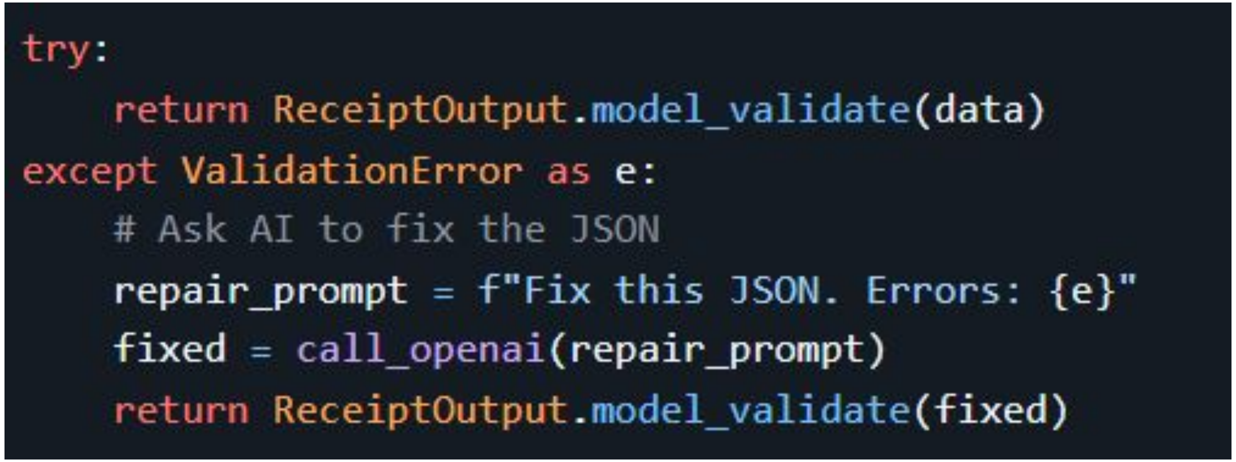
Pass the text data to OpenAI with a prompt to convert it to JSON

Pydantic

Used for data validation

Ensures the AI’s output matches our expected data

Validation comparison data is based on our earlier extracted Tesseract data



```
try:
    return ReceiptOutput.model_validate(data)
except ValidationError as e:
    # Ask AI to fix the JSON
    repair_prompt = f"Fix this JSON. Errors: {e}"
    fixed = call_openai(repair_prompt)
    return ReceiptOutput.model_validate(fixed)
```

Streamlit

Python framework for self hosted server and website

Uses a combination of CSS and Python to create webpage elements

Uses the formatted JSON data to drive various element printouts

From there you can take the returned data and put it into the next step of your data transformation pipeline