**Plum Backend Task (Project -8)**

**Ngrok link :** <https://73c66840cc9e.ngrok-free.app>

**GitHub link :** <https://github.com/Surjay124/AmountExtractorPlumTask>

**Spring Ollama Amount Extractor**

**Architecture used for OCR**

Tesseract OCR Architeture used: Tesseract has a modular OCR architecture:

Preprocessing → 2. Layout Analysis → 3. Recognition (LSTM) → 4. Postprocessing → 5. Output

It’s not purely a single deep learning model but a hybrid pipeline with LSTM at the core for modern Tesseract.

**Start OCR microservice (python)**

cd python-ocr

python3 -m venv venv

source venv/bin/activate

pip install -r requirements.txt

python ocr\_service.py

# service listens on http://localhost:5000/ocr

**python-ocr/ocr\_service.py (sample OCR microservice)**

This project uses **Ollama** to enhance medical document processing by providing AI-powered text understanding and structured data extraction.

**Setup Ollama**

Install Ollama on your machine:

* **macOS:** brew install ollama
* **Windows:** Download from [Ollama website](https://ollama.com/)
* **Linux:** Follow official instructions

After downloading, open terminal and write

ollama run mistral

It will take time to download 4 GB of data

Then, start the Ollama server:

ollama serve

The server will run at <http://localhost:11434>.

**Run Spring Boot app**

mvn spring-boot:run

**Sample curl**

Upload image:

curl -X POST "http://localhost:8080/api/amount/extract/file" -F "file=@sample\_bill.jpg"

Send text:

curl -X POST "http://localhost:8080/api/amount/extract/text" -H "Content-Type: application/json" -d '"Total: INR 1200 | Paid: 1000 | Due: 200 | Discount: 10%"'

If you want to do it on working link: Upload image:

curl -X POST "https://73c66840cc9e.ngrok-free.app/api/amount/extract/file" -F "file=@sample\_bill.jpg"

Send text:

curl -X POST "https://73c66840cc9e.ngrok-free.app/api/amount/extract/text" -H "Content-Type: application/json" -d '"Total: INR 1200 | Paid: 1000 | Due: 200 | Discount: 10%"

Design notes:

* OCR microservice is kept separate for best library support (pytesseract).
* Ollama is used via its HTTP API /api/generate with prompt engineering to return JSON. The code extracts the first JSON object returned by the model and parses it.
* Robustness: The system includes multiple fallback paths. If Ollama fails, a simple heuristic fallback maps normalized amounts to likely types.
* Guardrails: endpoints return {"status":"no\_amounts\_found","reason":"..."} when no reliable amounts are discovered.

**Current Issues and Future Improvements**

* Since, we have used a free AI model of Spring AI, which is Ollama, it is currently facing difficulties to extract information, despite the project running smoothly.
* Can be used in future, wherein we won’t require bills made up of paper and we can integrate this functionality in the medical apps used by respective hospitals.