@host.today/ht-docker-ai

local ai, that can be run easily

- readme.md for @host.today/ht-docker-ai
- changelog.md for @host.today/ht-docker-ai

readme.md for @host.today/ht-docker-ai

Production-ready Docker images for state-of-the-art AI Vision-Language Models. Run powerful multimodal AI locally with GPU acceleration—**no cloud API keys required**.

> **Three VLMs, one registry.** From high-performance document OCR to GPT-4o-level vision understanding—pick the right tool for your task.

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

What's Included

Model	Parameters	Best For	API	Port	VRAM
MiniCPM-V 4.5	8B	General vision understanding, multi-image analysis	Ollama-compatible	11434	~9GB
Nanonets-OCR2-3B	~3B	Document OCR with semantic markdown, LaTeX, flowcharts	OpenAI-compatible	8000	~12-16GB
Qwen3-VL-30B	30B (A3B)	Advanced visual agents, code generation from images	Ollama-compatible	11434	~20GB

□□ Quick Reference: All Available Images

```
code.foss.global/host.today/ht-docker-ai:<tag>
```

Tag	Model	Runtime	Port	VRAM
`minicpm45v` / `latest`	MiniCPM-V 4.5	Ollama	11434	~9GB
`nanonets-ocr`	Nanonets-OCR2-3B	vLLM	8000	~12-16GB
`qwen3vl`	Qwen3-VL-30B-A3B	Ollama	11434	~20GB

□□ MiniCPM-V 4.5

A GPT-4o level multimodal LLM from OpenBMB—handles image understanding, OCR, multi-image analysis, and visual reasoning across **30+ languages**.

□ Key Features

- □□ **Multilingual:** 30+ languages supported
- □□ **Multi-image:** Analyze multiple images in one request
- □□ **Versatile:** Charts, documents, photos, diagrams
- ⚡ **Efficient:** Runs on consumer GPUs (9GB VRAM)

Quick Start

```
docker run -d \
  --name minicpm \
  --gpus all \
  -p 11434:11434 \
  -v ollama-data:/root/.ollama \
  code.foss.global/host.today/ht-docker-ai:minicpm45v
```

> □□ **Pro tip:** Mount the volume to persist downloaded models (~5GB). Without it, models re-download on every container start.

API Examples

List models:

```
curl http://localhost:11434/api/tags
```

Analyze an image:

```
curl http://localhost:11434/api/generate -d '{
  "model": "minicpm-v",
  "prompt": "What do you see in this image?",
  "images": ["<base64-encoded-image>"]
}'
```

Chat with vision:

```
curl http://localhost:11434/api/chat -d '{
  "model": "minicpm-v",
  "messages": [{
    "role": "user",
    "content": "Describe this image in detail",
    "images": ["<base64-encoded-image>"]
  }]
}'
```

Hardware Requirements

Mode	VRAM Required
int4 quantized	~9GB
Full precision (bf16)	~18GB

□□ Nanonets-OCR2-3B

The **latest Nanonets document OCR model** (October 2025 release)—based on Qwen2.5-VL-3B, fine-tuned specifically for document extraction with significant improvements over the original OCR-s.

□ Key Features

- □□ **Semantic output:** Tables → HTML, equations → LaTeX, flowcharts → structured markup
- □□ **Multilingual:** Inherits Qwen's broad language support
- □□ **30K context:** Handle large, multi-page documents
- □□ **OpenAI-compatible:** Drop-in replacement for existing pipelines
- □□ **Improved accuracy:** Better semantic tagging and LaTeX equation extraction vs. OCR-s

Quick Start

```
docker run -d \
  --name nanonets \
  --gpus all \
  -p 8000:8000 \
  -v hf-cache:/root/.cache/huggingface \
  code.foss.global/host.today/ht-docker-ai:nanonets-ocr
```

API Usage

```
curl http://localhost:8000/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "nanonets/Nanonets-OCR2-3B",
    "messages": [{
      "role": "user",
      "content": [
        {"type": "image_url", "image_url": {"url": "<base64>"}},
        {"type": "text", "text": "Extract the text from the above document as if you were reading it naturally. Return the tables in html format. Return the equations in LaTeX representation."}
      ]
    }],
```

```
    "temperature": 0.0,
    "max_tokens": 4096
  }'
```

Output Format

Nanonets-OCR2-3B returns markdown with semantic tags:

Element	Output Format
Tables	`<table>...</table>` (HTML)
Equations	`$...$` (LaTeX)
Images	`<img>description</img>`
Watermarks	`<watermark>OFFICIAL COPY</watermark>`
Page numbers	`<page_number>14</page_number>`
Flowcharts	Structured markup

Hardware Requirements

Config	VRAM
30K context (default)	~12-16GB
Speed	~3-8 seconds per page

□□ Qwen3-VL-30B-A3B

The **most powerful** Qwen vision model—30B parameters with 3B active (MoE architecture). Handles complex visual reasoning, code generation from screenshots, and visual agent capabilities.

□ Key Features

- □□ **256K context** (expandable to 1M tokens!)
- □□ **Visual agent capabilities** — can plan and execute multi-step tasks
- □□ **Code generation from images** — screenshot → working code
- □□ **State-of-the-art** visual reasoning

Quick Start

```
docker run -d \
  --name qwen3vl \
  --gpus all \
  -p 11434:11434 \
  -v ollama-data:/root/.ollama \
  code.foss.global/host.today/ht-docker-ai:qwen3vl
```

Then pull the model (one-time, ~20GB):

```
docker exec qwen3vl ollama pull qwen3-vl:30b-a3b
```

API Usage

```
curl http://localhost:11434/api/chat -d '{
  "model": "qwen3-vl:30b-a3b",
  "messages": [{
    "role": "user",
    "content": "Analyze this screenshot and write the code to recreate this UI",
    "images": ["<base64-encoded-image>"]
  }]
}'
```

Hardware Requirements

Requirement	Value
VRAM	~20GB (Q4_K_M quantization)
Context	256K tokens default

□□ Docker Compose

Run multiple VLMs together for maximum flexibility:

```yaml
services:
  # General vision tasks
  minicpm:
    image: code.foss.global/host.today/ht-docker-ai:minicpm45v
    ports:
      - "11434:11434"
    volumes:
      - ollama-data:/root/.ollama
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]
    restart: unless-stopped

  # Document OCR with semantic output
  nanonets:
    image: code.foss.global/host.today/ht-docker-ai:nanonets-ocr
    ports:
      - "8000:8000"
    volumes:
      - hf-cache:/root/.cache/huggingface
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]
    restart: unless-stopped

volumes:
  ollama-data:
  hf-cache:
```

⚙ Environment Variables

MiniCPM-V 4.5 & Qwen3-VL (Ollama-based)

Variable	Default	Description
`MODEL_NAME`	`minicpm-v`	Ollama model to pull on startup
`OLLAMA_HOST`	`0.0.0.0`	API bind address
`OLLAMA_ORIGINS`	`*`	Allowed CORS origins

Nanonets-OCR (vLLM-based)

Variable	Default	Description
`MODEL_NAME`	`nanonets/Nanonets-OCR2-3B`	HuggingFace model ID
`HOST`	`0.0.0.0`	API bind address
`PORT`	`8000`	API port
`MAX_MODEL_LEN`	`30000`	Maximum sequence length
`GPU_MEMORY_UTILIZATION`	`0.9`	GPU memory usage (0-1)

□□ Architecture Notes

Dual-VLM Consensus Strategy

For production document extraction, consider using multiple models together:

1. **Pass 1:** MiniCPM-V visual extraction (images → JSON)
2. **Pass 2:** Nanonets-OCR semantic extraction (images → markdown → JSON)
3. **Consensus:** If results match → Done (fast path)
4. **Pass 3+:** Additional visual passes if needed

This dual-VLM approach catches extraction errors that single models miss.

Why Multi-Model Works

- **Different architectures:** Independent models cross-validate each other
- **Specialized strengths:** Nanonets-OCR2-3B excels at document structure; MiniCPM-V handles general vision
- **Native processing:** All VLMs see original images—no intermediate structure loss

Model Selection Guide

Task	Recommended Model
General image understanding	MiniCPM-V 4.5
Document OCR with structure preservation	Nanonets-OCR2-3B
Complex visual reasoning / code generation	Qwen3-VL-30B
Multi-image analysis	MiniCPM-V 4.5
Visual agent tasks	Qwen3-VL-30B
Large documents (30K+ tokens)	Nanonets-OCR2-3B

□□ Building from Source

```
# Clone the repository
git clone https://code.foss.global/host.today/ht-docker-ai.git
cd ht-docker-ai

# Build all images
./build-images.sh

# Run tests
pnpm test
```

□□ Troubleshooting

Model download hangs

```
docker logs -f <container-name>
```

Model downloads can take several minutes (~5GB for MiniCPM-V, ~20GB for Qwen3-VL).

Out of memory

- **GPU:** Use a lighter model variant or upgrade VRAM
- **CPU:** Increase container memory: `--memory=16g`

API not responding

1. Check container health: `docker ps`
2. Review logs: `docker logs <container>`
3. Verify port: `curl localhost:11434/api/tags` or `curl localhost:8000/health`

Enable NVIDIA GPU support on host

```
# Install NVIDIA Container Toolkit
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o
/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-
toolkit.list | \
  sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg]
https://#g' | \
  sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
sudo apt-get update && sudo apt-get install -y nvidia-container-toolkit
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

GPU Memory Contention (Multi-Model)

When running multiple VLMs on a single GPU:

- vLLM and Ollama both need significant GPU memory
- **Single GPU:** Run services sequentially (stop one before starting another)
- **Multi-GPU:** Assign each service to a different GPU via `CUDA_VISIBLE_DEVICES`

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @host.today/ht-docker-ai

2026-01-20 - 1.16.0 - feat(invoices)

add line_items extraction and normalization for invoice parsing

- Introduce ILineItem interface and add line_items array to IInvoice.
- Add extractLineItems helper to normalize item fields (position, product, description, quantity, unit_price, total_price).
- Include line_items in parsed invoice output and sample JSON in test, defaulting to [] when absent.
- Update logging to include extracted line item count.
- Clarify test instructions to extract items from invoice tables and skip subtotal/total rows.

2026-01-20 - 1.15.3 - fix(tests(nanonets))

allow '/' when normalizing invoice strings in tests

- Adjust regex in test/test.invoices.nanonets.ts to preserve forward slashes when cleaning invoice values
- Changed pattern from [^A-Z0-9-] to [^A-Z0-9/-] to prevent accidental removal of '/' characters in invoice identifiers

2026-01-20 - 1.15.2 - fix(dev-deps)

bump devDependencies @push.rocks/smartagent to ^1.6.2 and @push.rocks/smartai to ^0.13.3

- Bumped @push.rocks/smartagent from ^1.5.4 to ^1.6.2 in devDependencies
- Bumped @push.rocks/smartai from ^0.13.2 to ^0.13.3 in devDependencies
- Updated test/test.invoices.nanonets.ts JSON extraction prompt: instruct not to omit special characters in invoice_number and to use the json validate tool
- No breaking changes; only dev dependency updates and test prompt adjustments

2026-01-20 - 1.15.1 - fix(tests)

enable progress events in invoice tests and bump @push.rocks/smartagent devDependency to ^1.5.4

- Added an onProgress handler in test/test.invoices.nanonets.ts to log progress events (console.log(event.logMessage)) so tool calls and progress are visible during tests.
- Bumped devDependency @push.rocks/smartagent from ^1.5.2 to ^1.5.4 in package.json.

2026-01-20 - 1.15.0 - feat(tests)

integrate SmartAi/DualAgentOrchestrator into extraction tests and add JSON self-validation

- Integrate SmartAi and DualAgentOrchestrator into bankstatement and invoice tests to perform structured extraction with streaming
- Register and use JsonValidatorTool to validate outputs (json.validate) and enforce validation before task completion
- Add tryExtractJson parsing fallback, improved extraction prompts, retries and clearer parsing/logging
- Initialize and teardown SmartAi and orchestrator in test setup/summary, and enable onToken streaming handlers for real-time output
- Bump devDependencies: @push.rocks/smartagent to ^1.3.0 and @push.rocks/smartai to ^0.12.0

2026-01-20 - 1.14.3 - fix(repo)

no changes detected in the diff; no files modified and no release required

- Diff contained no changes
- No files were added, removed, or modified
- No code, dependency, or documentation updates to release

2026-01-19 - 1.14.2 - fix(readme)

update README to document Nanonets-OCR2-3B (replaces Nanonets-OCR-s), adjust VRAM and context defaults, expand feature docs, and update examples/test command

- Renamed Nanonets-OCR-s -> Nanonets-OCR2-3B throughout README and examples
- Updated Nanonets VRAM guidance from ~10GB to ~12-16GB and documented 30K context
- Changed documented MAX_MODEL_LEN default from 8192 to 30000
- Updated example model identifiers (model strings and curl/example snippets) to nanonets/Nanonets-OCR2-3B
- Added MiniCPM and Qwen feature bullets (multilingual, multi-image, flowchart support, expanded context notes)
- Replaced README test command from ./test-images.sh to pnpm test

2026-01-19 - 1.14.1 - fix(extraction)

improve JSON extraction prompts and model options for invoice and bank statement tests

- Refactor JSON extraction prompts to be sent after the document text and add explicit 'WHERE TO FIND DATA' and 'RULES' sections for clearer extraction guidance
- Change chat message flow to: send document, assistant acknowledgement, then the JSON extraction prompt (avoids concatenating large prompts into one message)
- Add model options (num_ctx: 32768, temperature: 0) to give larger context windows and deterministic JSON output
- Simplify logging to avoid printing full prompt contents; log document and prompt lengths instead
- Increase timeouts for large documents to 600000ms (10 minutes) where applicable

2026-01-19 - 1.14.0 - feat(docker-images)

add vLLM-based Nanonets-OCR2-3B image, Qwen3-VL Ollama image and refactor build/docs/tests to use new runtime/layout

- Add new Dockerfiles for Nanonets (Dockerfile_nanonets_vllm_gpu_VRAM10GB), Qwen3 (Dockerfile_qwen3vl_ollama_gpu_VRAM20GB) and a clarified MiniCPM Ollama variant (Dockerfile_minicpm45v_ollama_gpu_VRAM9GB); remove older, redundant Dockerfiles.
- Update build-images.sh to build the new image tags (minicpm45v, qwen3vl, nanonets-ocr) and adjust messaging/targets accordingly.
- Documentation overhaul: readme.md and readme.hints.md updated to reflect vLLM vs Ollama runtimes, corrected ports/VRAM estimates, volume recommendations, and API endpoint details.
- Tests updated to target the new model ID (nanonets/Nanonets-OCR2-3B), to process one page per batch, and to include a 10-minute AbortSignal timeout for OCR requests.
- Added focused extraction test suites (test/test.invoices.extraction.ts and test/test.invoices.failed.ts) for faster iteration and debugging of invoice extraction.
- Bump devDependencies: @git.zone/tsrun -> ^2.0.1 and @git.zone/tstest -> ^3.1.5.
- Misc: test helper references and docker compose/test port mapping fixed (nanonets uses 8000), and various README sections cleaned and reorganized.

2026-01-18 - 1.13.2 - fix(tests)

stabilize OCR extraction tests and manage GPU containers

- Add stopAllGpuContainers() and call it before starting GPU images to free GPU memory.
- Remove PaddleOCR-VL image configs and associated ensure helpers from docker test helper to simplify images list.
- Split invoice/bankstatement tests into two sequential stages: Stage 1 runs Nanonets OCR to produce markdown files, Stage 2 stops Nanonets and runs model extraction from saved markdown (avoids GPU contention).
- Introduce temporary markdown directory handling and cleanup; add stopNanonets() and container running checks in tests.
- Switch bank statement extraction model from qwen3:8b to gpt-oss:20b; add request timeout and improved logging/console output across tests.
- Refactor extractWithConsensus and extraction functions to accept document identifiers, improve error messages and JSON extraction robustness.

2026-01-18 - 1.13.1 - fix(image_support_files)

remove PaddleOCR-VL server scripts from image_support_files

- Deleted files: image_support_files/paddleocr_vl_full_server.py (approx. 636 lines) and image_support_files/paddleocr_vl_server.py (approx. 465 lines)
- Cleanup/removal of legacy PaddleOCR-VL FastAPI server implementations — may affect users who relied on these local scripts

2026-01-18 - 1.13.0 - feat(tests)

revamp tests and remove legacy Dockerfiles: adopt JSON/consensus workflows, switch MiniCPM model, and delete deprecated Docker/test variants

- Removed multiple Dockerfiles and related entrypoints for MiniCPM and PaddleOCR-VL (cpu/gpu/full), cleaning up legacy image recipes.
- Pruned many older test files (combined, ministral3, paddleocr-vl, and several invoice/test variants) to consolidate the test suite.
- Updated bank statement MiniCPM test: now uses MODEL='openbmb/minicpm-v4.5:q8_0', JSON per-page extraction prompt, consensus retry logic, expanded logging, and stricter result matching.
- Updated invoice MiniCPM test: switched to a consensus flow (fast JSON pass + thinking pass), increased PDF conversion quality, endpoints migrated to chat-style API calls with image-in-message payloads, and improved finalization logic.
- API usage changed from /api/generate to /api/chat with message-based payloads and embedded images — CI and local test runners will need model availability and possible pipeline adjustments.

2026-01-18 - 1.12.0 - feat(tests)

switch vision tests to multi-query extraction (count then per-row/field queries) and add logging/summaries

- Replace streaming + consensus pipeline with multi-query approach: count rows per page, then query each transaction/field individually (batched parallel queries).
- Introduce unified helpers (queryVision / queryField / getTransaction / countTransactions) and simplify Ollama requests (stream:false, reduced num_predict, /no_think prompts).
- Improve parsing and normalization for amounts (European formats), invoice numbers, dates and currency extraction.
- Adjust model checks to look for generic 'minicpm' and update test names/messages; add pass/fail counters and a summary test output.
- Remove previous consensus voting and streaming JSON accumulation logic, and add immediate per-transaction logging and batching.

2026-01-18 - 1.11.0 - feat(vision)

process pages separately and make Qwen3-VL vision extraction more robust; add per-page parsing, safer JSON handling, reduced token usage, and multi-query invoice extraction

- Bank statements: split extraction into extractTransactionsFromPage and sequentially process pages to avoid thinking-token exhaustion
- Bank statements: reduced num_predict from 8000 to 4000, send single image per request, added per-page logging and non-throwing handling for empty or non-JSON responses
- Bank statements: catch JSON.parse errors and return empty array instead of throwing
- Invoices: introduced queryField to request single values and perform multiple simple queries (reduces model thinking usage)
- Invoices: reduced num_predict for invoice queries from 4000 to 500 and parse amounts robustly (handles European formats like 1.234,56)
- Invoices: normalize currency to uppercase 3-letter code, return safe defaults (empty strings / 0) instead of nulls, and parse net/vat/total with fallbacks
- General: simplified Ollama API error messages to avoid including response body content in thrown errors

2026-01-18 - 1.10.1 - fix(tests)

improve Qwen3-VL invoice extraction test by switching to non-stream API, adding model availability/pull checks, simplifying response parsing, and tightening model options

- Replaced streaming reader logic with direct JSON parsing of the /api/chat response
- Added ensureQwen3Vl() to check and pull the Qwen3-VL:8b model from Ollama
- Switched to ensureMiniCpm() to verify Ollama service is running before model checks
- Use /no_think prompt for direct JSON output and set temperature to 0.0 and num_predict to 512
- Removed retry loop and streaming parsing; improved error messages to include response body
- Updated logging and test setup messages for clarity

2026-01-18 - 1.10.0 - feat(vision)

add Qwen3-VL vision model support with Dockerfile and tests; improve invoice OCR conversion and prompts; simplify extraction flow by removing consensus voting

- Add Dockerfile_qwen3vl to provide an Ollama-based image for Qwen3-VL and expose the Ollama API on port 11434
- Introduce test/test.invoices.qwen3vl.ts and ensureQwen3Vl() helper to pull and test qwen3-vl:8b
- Improve PDF->PNG conversion and prompt in ministral3 tests (higher DPI, max quality, sharpen) and increase num_predict from 512 to 1024
- Simplify extraction pipeline: remove consensus voting, log single-pass results, and simplify OCR HTML sanitization/truncation logic

2026-01-18 - 1.9.0 - feat(tests)

add Ministral 3 vision tests and improve invoice extraction pipeline to use Ollama chat schema, sanitization, and multi-page support

- Add new vision-based test suites for Ministral 3: test/test.invoices.ministral3.ts and test/test.bankstatements.ministral3.ts (model ministral-3:8b).
- Introduce ensureMinistral3() helper to start/check Ollama/MiniCPM model in test/helpers/docker.ts.
- Switch invoice extraction to use Ollama /api/chat with a JSON schema (format) and streaming support (reads message.content).
- Improve HTML handling: sanitizeHtml() to remove OCR artifacts, concatenate multi-page HTML with page markers, and increase truncation limits.
- Enhance response parsing: strip Markdown code fences, robustly locate JSON object boundaries, and provide clearer JSON parse errors.
- Add PDF->PNG conversion (ImageMagick) and direct image-based extraction flow for vision model tests.

2026-01-18 - 1.8.0 - feat(paddleocr-vl)

add structured HTML output and table parsing for PaddleOCR-VL, update API, tests, and README

- Add result_to_html(), parse_markdown_table(), and parse_paddleocr_table() to emit semantic HTML and convert OCR/markdown tables to proper

- Enhance result_to_markdown() with positional/type hints (header/footer/title/table/figure) to improve downstream LLM processing
- Expose 'html' in supported formats and handle output_format='html' in parse endpoints and CLI flow

- Update tests to request HTML output and extract invoice fields from structured HTML (test/test.invoices.paddleocr-vl.ts)
- Refresh README with usage, new images/tags, architecture notes, and troubleshooting for the updated pipeline

2026-01-17 - 1.7.1 - fix(docker)

standardize Dockerfile and entrypoint filenames; add GPU-specific Dockerfiles and update build and test references

- Added Dockerfile_minicpm45v_gpu and image_support_files/minicpm45v_entrypoint.sh; removed the old Dockerfile_minicpm45v and docker-entrypoint.sh
- Renamed and simplified PaddleOCR entrypoint to image_support_files/paddleocr_vl_entrypoint.sh and updated CPU/GPU Dockerfile references
- Updated build-images.sh to use *_gpu Dockerfiles and clarified PaddleOCR GPU build log
- Updated test/helpers/docker.ts to point to Dockerfile_minicpm45v_gpu so tests build the GPU variant

2026-01-17 - 1.7.0 - feat(tests)

use Qwen2.5 (Ollama) for invoice extraction tests and add helpers for model management; normalize dates and coerce numeric fields

- Added ensureOllamaModel and ensureQwen25 test helpers to pull/check Ollama models via localhost:11434
- Updated invoices test to use qwen2.5:7b instead of MiniCPM and removed image payload from the text-only extraction step
- Increased Markdown truncate limit from 8000 to 12000 and reduced model num_predict from 2048 to 512
- Rewrote extraction prompt to require strict JSON output and added post-processing to parse/convert numeric fields
- Added normalizeDate and improved compareInvoice to normalize dates and handle numeric formatting/tolerance
- Updated test setup to ensure Qwen2.5 is available and adjusted logging/messages to reflect the Qwen2.5-based workflow

2026-01-17 - 1.6.0 - feat(paddleocr-vl)

add PaddleOCR-VL full pipeline Docker image and API server, plus integration tests and docker helpers

- Add Dockerfile_paddleocr_vl_full and entrypoint script to build a GPU-enabled image with PP-DocLayoutV2 + PaddleOCR-VL and a FastAPI server
- Introduce image_support_files/paddleocr_vl_full_server.py implementing the full pipeline API (/parse, OpenAI-compatible /v1/chat/completions) and a /formats endpoint
- Improve image handling: decode_image supports data URLs, HTTP(S), raw base64 and file paths; add optimize_image_resolution to auto-scale images into the recommended 1080-2048px range
- Add test helpers (test/helpers/docker.ts) to build/start/health-check Docker images and new ensurePaddleOcrVlFull workflow
- Add comprehensive integration tests for bank statements and invoices (MiniCPM and PaddleOCR-VL variants) and update tests to ensure required containers are running before tests
- Switch MiniCPM model references to 'minicpm-v:latest' and increase health/timeout expectations for the full pipeline

2026-01-17 - 1.5.0 - feat(paddleocr-vl)

add PaddleOCR-VL GPU Dockerfile, pin vllm, update CPU image deps, and improve entrypoint and tests

- Add a new GPU Dockerfile for PaddleOCR-VL (transformers-based) with CUDA support, healthcheck, and entrypoint.
- Pin vllm to 0.11.1 in Dockerfile_paddleocr_vl to use the first stable release with PaddleOCR-VL support.
- Update CPU image: add torchvision==0.20.1 and extra Python deps (protobuf, sentencepiece, einops) required by the transformers-based server.
- Rewrite paddleocr-vl-entrypoint.sh to build vllm args array, add MAX_MODEL_LEN and ENFORCE_EAGER env vars, include --limit-mm-per-prompt and optional --enforce-eager, and switch to exec vllm with constructed args.
- Update tests to use the OpenAI-compatible PaddleOCR-VL chat completions API (/v1/chat/completions) with image+text message payload and model 'paddleocr-vl'.

- Add @types/node to package.json dependencies and tidy devDependencies ordering.

2026-01-16 - 1.4.0 - feat(invoices)

add hybrid OCR + vision invoice/document parsing with PaddleOCR, consensus voting, and prompt/test refactors

- Add hybrid pipeline documentation and examples (PaddleOCR + MiniCPM-V) and architecture diagram in recipes/document.md
- Integrate PaddleOCR: new OCR extraction functions and OCR-only prompt flow in test/test.node.ts
- Add consensus voting and parallel-pass optimization to improve reliability (multiple passes, hashing, and majority voting)
- Refactor prompts and tests: introduce /nothink token, OCR truncation limits, separate visual and OCR-only prompts, and improved prompt building in test/test.invoices.ts
- Update image conversion defaults (200 DPI, filename change) and add TypeScript helper functions for extraction and consensus handling

2026-01-16 - 1.3.0 - feat(paddleocr)

add PaddleOCR OCR service (Docker images, server, tests, docs) and CI workflows

- Add GPU and CPU PaddleOCR Dockerfiles; pin paddlepaddle/paddle and paddleocr to stable 2.x and install libgomp1 for CPU builds
- Avoid pre-downloading OCR models at build-time to prevent build-time segfaults; models are downloaded on first run
- Refactor PaddleOCR FastAPI server: respect CUDA_VISIBLE_DEVICES, support per-request language, cache default language instance and create temporary instances for other languages
- Add comprehensive tests (test.paddleocr.ts) and improve invoice extraction tests (parallelize passes, JSON OCR API usage, prioritize certain test cases)
- Add Gitea CI workflows for tag and non-tag Docker runs and release pipeline (docker build/push, metadata trigger)
- Update documentation (readme.hints.md) with PaddleOCR usage and add docker registry entry to npmextra.json

2026-01-16 - 1.2.0 - feat(paddleocr)

add PaddleOCR support: Docker images, FastAPI server, entrypoint and tests

- Add PaddleOCR FastAPI server implementation at image_support_files/paddleocr_server.py
- Remove old image_support_files/paddleocr-server.py and update entrypoint to import paddleocr_server:app
- Extend build-images.sh to build paddleocr (GPU) and paddleocr-cpu images and list them
- Extend test-images.sh to add paddleocr health/OCR tests, new test_paddleocr_image function, port config, and cleanup; rename test_image -> test_minicpm_image

2026-01-16 - 1.1.0 - feat(ocr)

add PaddleOCR GPU Docker image and FastAPI OCR server with entrypoint; implement OCR endpoints and consensus extraction testing

- Add Dockerfile_paddleocr for GPU-accelerated PaddleOCR image (pre-downloads PP-OCRv4 models, exposes port 5000, healthcheck, entrypoint)
- Add image_support_files/paddleocr-server.py: FastAPI app providing /ocr (base64), /ocr/upload (file), and /health endpoints; model warm-up on startup; structured JSON responses and error handling
- Add image_support_files/paddleocr-entrypoint.sh to configure environment, detect GPU/CPU mode, and launch uvicorn
- Update test/test.node.ts to replace streaming extraction with a consensus-based extraction flow (multiple passes, hashing of results, majority voting) and improve logging/prompt text
- Add test/test.invoices.ts: integration tests for invoice extraction that call PaddleOCR, build prompts with optional OCR text, run consensus extraction, and produce a summary report

2026-01-16 - 1.0.0 - initial release

Initial project files added with two small follow-up updates.

- initial: base project commit.
- update: two minor follow-up updates refining the initial commit.