LLM-Judge Fake Receipt Detector (Take-Home)

1) Project goal

Build a tool that uses **multiple LLM "judges"** to **vote** whether a receipt is **FAKE** or **REAL**, with short rationales.

Tooling freedom (important):

- You may use **AI coding assistants** (e.g., Copilot, Cursor, ChatGPT, etc.) and **any LLM APIs/models**.
- There are **no limitations** on which tools/models you choose.
- You must **share and explain** how you used these tools (what you used them for, what worked, what didn't). We're interested in your **experience and judgment** using AI tools as part of the workflow.

Expected explanation (keep it lightweight):

- **README length**: aim for ~1–2 pages (bullet points are fine). No need for a detailed daily log.
- **Design explanation**: 1–2 short paragraphs on your architecture and key decisions.
- **Prompting**: include the core judge prompt(s) you used (copy/paste is fine) plus 2–3 sentences on why you structured them that way.
- **AI tools usage**: a short section (a few bullets) on how you used coding assistants/LLMs during development.
- **Evidence**: include at least one example run (input receipt filename/ID + judges' JSON outputs + final vote).

Demo flow (must-have):

1) Explore and present a quick **dataset distribution** summary (notebook + presentation)
2) Load a curated **20-receipt evaluation set**
3) Pick a receipt → view image + extracted fields
4) Run **3 judges** → see each judge's JSON decision + rationale

5) Aggregated decision by **voting** + explanation tags

---

2) Dataset (open source)

Recommended dataset: **Find it again! – Receipt Dataset for Document Forgery Detection**

Use it as the receipt dataset with **images + ground-truth labels** for **authentic (real)** vs **forged (fake)**.

Dataset links - Dataset page: https://l3i-share.univ-lr.fr/2023Finditagain/index.html - Direct download (ZIP): https://l3i-share.univ-lr.fr/2023Finditagain/findit2.zip

What's inside (high level) - **988 receipt images** sourced from SROIE - **163 realistically forged** receipts + authentic receipts - Split files (e.g., train/val/test lists) include **filenames + ground truth labels**

Why it fits - Already labeled **FAKE vs REAL** (no need to synthesize fakes) - Receipt-specific for forgery detection - Small enough for a 2–3 day take-home

> You can still add optional synthetic tampering as an extension, but it's not required.

---

3) Dataset distribution exploration (candidate must present)

The candidate must explore the dataset and present a short summary of its distribution. Keep this lightweight but concrete.

Must include (at minimum):

- Count of **REAL vs FAKE** receipts
- Distribution of **receipt totals** (histogram or binned bar chart)
- Any additional insights they can extract from the dataset (examples below)

Examples of optional insights (pick any):

- Totals by label (REAL vs FAKE) - are fakes skewed higher/lower?
- Image size / resolution distribution (width×height)
- File size distribution (KB/MB) as a rough proxy for compression/quality
- Aspect ratio patterns (e.g., long receipts vs shorter ones)

- Simple image-quality proxies (blur/sharpness via variance of Laplacian; brightness/contrast)

Where to show it:

- **Notebook** (recommended): a short exploration notebook that generates the charts/tables
- **Presentation**: summarize the key findings in **1 slide** (or a short section) - *(Optional)* In the app: a small chart is a nice extra, but not required

---

4) "20 sampling" requirement

Randomly select **20 receipts**: **10 REAL + 10 FAKE**.

Candidate requirement: - Implement the sampling script (random seed recommended) - Output the list of selected filenames/IDs and labels

What they must include in the README: - The random seed (if used) and exact selection method - The final list of the 20 selected receipt IDs/filenames and their labels

5) LLM(s) as judges + voting

Use **3 judges** (either 3 different models or same model with different "judge personas" and/or temperatures).

Judge input

For each receipt, provide: - **Receipt image (required)** — the judge decision must be based primarily on the image. - *(Optional)* Minimal metadata: filename/ID and the dataset label (for evaluation only; **do not** show the ground-truth label to judges).

> No OCR is required. If you choose to add extracted fields later, treat it as an optional enhancement.

Judge output (JSON, extensible)

Each judge must return **valid JSON** with at least these required fields:

```
{
  "label": "FAKE|REAL|UNCERTAIN",
  "confidence": 0.0, // confidence Range [0.0-100.]
  "reasons": ["short reason 1", "short reason 2"]
}
```

Optional (encouraged) extensions: candidates may add additional fields if they have a better idea (e.g., `flags`, `evidence_regions`, `risk_level`, `calibration_notes`, etc.). If they extend the schema, they should document it briefly in the README.

Voting logic (simple)

- Run **3 judges**.
- Take a **majority vote**:
  - If **2 or 3** judges say **FAKE** → final = **FAKE**
  - If **2 or 3** judges say **REAL** → final = **REAL**
  - Otherwise (no majority, or too many **UNCERTAIN**) → final = **UNCERTAIN**

What to display: - The final label + vote tally (e.g., `FAKE (2/3)`). - Each judge's `confidence` and `reasons` (no extra aggregation required).

> **Optional extension:** You can propose a better vote aggregation mechanism (e.g., confidence-weighted voting or tie-break rules) if you want—just keep it simple, clearly explained, and easy to follow.

UI presentation

Show: - Final vote + confidence summary - Per-judge JSON + rationales side-by-side - Highlighted flags (chips/tags)

---

6) Interactive app (optional extension)

A UI is not required.

✅ A clean notebook-only solution is fully acceptable (and often preferred) — please do not feel pressure to build a UI.

If you have time, you may build a simple interactive app (Streamlit / Next.js / Flask + basic HTML) to make the demo easier.

Suggested screens/components (optional):

- Receipt browser: list of the 20 samples
- Receipt viewer: image + parsed fields
- "Run judges" button
- Results panel: per-judge outputs + final vote

Suggested quality items (optional): - Valid JSON enforcement (retry or repair on invalid JSON) - Basic error handling (timeouts, missing fields)

---

7) What we're evaluating

This is not a classical ML modeling task. Please don't over-engineer.

We care most about:

- **System design clarity**: clean architecture, simple pipeline, clear interfaces
- **LLM usage judgment**: how you prompt, how you handle uncertainty, and why you chose your approach/models
- **Evaluation rigor**: sensible evaluation on the labeled data + a thoughtful error analysis
- **Code quality**: readability, structure, reproducibility, and basic robustness
- **Communication**: clear README + clear presentation of decisions/tradeoffs (including how you used AI tools)

8) Deliverables

1) Working solution (**CLI or notebook**) that runs end-to-end
2) Short notebook for **dataset distribution** exploration + 1 slide/section in the presentation
1) Data pipeline:
    - download/load dataset
    - randomly sample 20 receipts (10 REAL / 10 FAKE) and output the final list
3) Evaluation summary:
    - accuracy on your labeled REAL/FAKE set
    - 2–3 examples where judges disagree and why
4) README (lightweight):
    - architecture summary (1–2 paragraphs)
    - **judge prompt(s)** used (copy/paste)
    - how you used AI tools / coding assistants
    - at least one sample run (judges' JSON + final vote)
5) Screenshots/GIF (recommended, not required):
    - UI screenshots if you build a UI, otherwise a sample CLI/notebook output