Take-Home Task — Tauri PDF Summarizer

Goal

Build a small installable Tauri desktop chatbot that ingests a PDF, summarizes it into structured bullet points, and highlights the items government officials in India care about (procurement deadlines, tender value, eligibility, key deliverables, penalties, contact points). Keep it minimal and finishable in 1 to 2 days. Feel free to use AI to build it. Hence forth code quality and project structure will also be taken into account.

Core requirements (must)

1. Tauri app (Rust backend + web frontend).

2. Open a local PDF (file picker) and extract text. (PDF text extraction may be done in frontend with pdfjs or in Rust.)

3. Summarize PDF into a structured bulleted output with these fields for each section/point:

    - short_summary (1–2 lines)

    - relevance_to_officials (2–4 bullets: deadlines, money, eligibility, actions)

    - action_items (bullet list)

    - confidence_estimate (low/medium/high or numeric)

4. Use a freely available summarization pathway:

    - **Primary (easy):** call a public/free inference endpoint (Hugging Face inference API or other open endpoints). Provide instructions how to supply an API key but app should run in mock mode without a key.

    - **Offline fallback (required):** include a mock summarizer (deterministic rule-based or small local summarizer) so the app works without any external keys.

5. Minimal conversational UI:

    - left: PDF viewer or filename + page selector

    - center: chat-like pane where user asks (e.g., “Summarize for procurement officer”) and sees structured output
    - right: raw extracted text (collapsible)
6. Packaging / installability: include instructions and a single script to build (tauri dev and a simple tauri build or packaging command). Candidate must show the app runs locally (dev mode is acceptable if packaging is heavy).

Optional (nice to have)

- Auto-detect sections that look like dates, money, eligibility, contact details and highlight them.
- Export summary as Markdown or CSV.
- A small preset prompt “For Government Official: focus on procurement, deadlines, thresholds” with one-click apply.
- Cross-platform packaged builds (mac/linux/windows) or one packaged artifact for the platform used.

Minimal tech hints (suggested)

- Tauri (Rust) + frontend: React + Vite (or Svelte).
- PDF text extraction: pdfjs-dist in frontend OR poppler/mupdf bindings in Rust. Frontend pdfjs is simplest.
- Summarizer options:
    - Call Hugging Face Inference API (model: summarization-capable). Explain how to plug API token.
    - Or call local LLM via llama.cpp if candidate knows it (optional).
    - Provide a deterministic mock summarizer: extract first N sentences + regex detect numbers/dates/money and tag.
- IPC: frontend sends extracted text to Rust for processing (or directly calls API from frontend — either is fine).

- Storage: none required. Temporary in-memory only.

Deliverables (what to submit)

1. GitHub repo link with source.
2. README with 1-line app summary and exact run steps:
   - dev run (npm + tauri dev)
   - build/pack (tauri build) or packaging hint
   - how to run mock mode vs real API mode (env vars)
3. Short demo (GIF or 1–2 minute video) showing: open PDF → ask “Summarize for procurement officer” → structured output.
4. A sample test PDF (one or two government tender PDFs or simulated tender text). If real tender URLs are not included, provide a small sample PDF included in repo.
5. Minimal tests (unit test for the mock summarizer).

Acceptance Criteria (automatic check)

- App opens and user can pick a PDF.
- App extracts text and returns structured JSON with fields: short_summary, relevance_to_officials, action_items, confidence_estimate.
- README shows how to run in mock mode without API keys.
- Demo GIF/video included.

Evaluation rubric (concise)

- Functionality 40% — PDF in → structured summary out, mock mode works, fields present.
- UX & clarity 20% — clear UI, instructions, easy to use for non-technical official.
- Code quality & tests 15% — readable code, one or two unit tests, clear repo structure.

- Docs & deploy 15% — README, build instructions, demo included.
- Domain fit & creativity 10% — relevance to gov official needs (deadlines, money, eligibility highlighted).

Scoring notes for graders

- If API mode is broken but mock mode works and README shows API instructions, still pass core functionality.
- If extraction fails for scanned PDFs (images), still acceptable if README documents limitation and candidate suggests OCR path (tesseract) as next step.

Small sample prompt for the model (candidate can use as default)

"You are a summarizer for Indian government procurement officers. From the provided document extract the most important bullets an officer needs to act on: procurement value, submission deadline(s), eligibility criteria, required documents, penalties, key contacts, and suggested next steps. Return JSON with keys short_summary, relevance_to_officials (array), action_items (array), confidence_estimate."

Quick checklist to submit

- Repo link
- README with run + build + mock instructions
- Demo GIF/video (<=2min)
- Sample PDF(s) included
- Passing unit test for mock summarizer