Task 1 — "Court-Data Fetcher & Mini-Dashboard"

Objective

Build a small web app that lets a user choose a **Case Type** and **Case Number** for a specific Indian court, then fetches and displays the case metadata and latest orders/judgments.

Court to target (example)

- **Delhi High Court** (https://delhihighcourt.nic.in/) **or**
- Any single **District-Court eCourts portal** (e.g., *Faridabad District Court* on https://districts.ecourts.gov.in/).

  Pick whichever you can reliably scrape within time; declare the choice in your README.

Functional requirements

1. **UI** – simple form (HTML or lightweight framework) with dropdown/inputs: *Case Type*, *Case Number*, *Filing Year*.
2. **Backend** – on submission, programmatically request the public court site, bypass any view-state tokens/CAPTCHA (document your approach), and parse:
   - Parties' names
   - Filing & next-hearing dates
   - Order/judgment PDF links (show at least the most recent one)
3. **Storage** – log each query & raw response in SQLite / PostgreSQL (any RDBMS).
4. **Display** – render the parsed details nicely; allow downloading any linked PDFs.
5. **Error handling** – user-friendly messages for invalid case numbers or site downtime.

Deliverables

Item	**Details**
Code repo	Public GitHub with MIT or Apache-2.0 licence
README	Court chosen, setup steps, CAPTCHA strategy, sample env vars

Item	Details
Demo video	≤ 5 min screen-capture showing end-to-end flow
Optional extras	Dockerfile, pagination for multiple orders, simple unit tests, CI workflow

Tech freedom

Use any language or stack you love (Python/Flask, Node/Express, Go, etc.). Feel free to leverage **Playwright, Selenium, or headless-browser AI helpers (Claude Code, GPT-4o)** for navigation.

Evaluation hints (for you, the interviewer)

- Robustness against site layout changes
- Code clarity & security (no hard-coded secrets)
- Documentation quality
- UI/UX simplicity
- Creativity in circumventing CAPTCHA legally (e.g., remote CAPTCHA-solving service, manual token field, or court API if available)

Task 2 — "WhatsApp-Driven Google Drive Assistant (n8n workflow)"

Objective

Create an **n8n** workflow that listens to WhatsApp messages and performs Google Drive actions such as listing files, deleting, moving, and summarising documents in a folder.

Functional requirements

1. **Inbound channel**
   - Use **Twilio Sandbox for WhatsApp** (or any WhatsApp Cloud API) as the entry point.
   - Parse simple text commands, e.g.:
     - LIST /ProjectX — list files in /ProjectX
     - DELETE /ProjectX/report.pdf
     - MOVE /ProjectX/report.pdf /Archive
     - SUMMARY /ProjectX — return a bullet digest of each file's content (PDF/Docx/TXT).
2. **Google Drive integration**
   - OAuth2 connection; operate only within the authenticated user's drive.
   - Use MIME-type aware nodes for reading content.
3. **AI summarisation**
   - Call **OpenAI GPT-4o** (or Claude) inside n8n to produce concise summaries.
4. **Logging & safety**
   - Maintain an audit spreadsheet/log.
   - Guard against accidental mass deletion (e.g., require confirmation keyword).
5. **Deployment**
   - Provide a ready-to-import **workflow.json** plus instructions for running n8n via Docker.

Deliverables

Item	**Details**
GitHub repo	Contains workflow JSON, env-sample, helper scripts
README	Setup of Twilio sandbox, Google credentials, command syntax
Demo video	≤ 5 min showing WhatsApp conversations triggering Drive actions
Optional extras	Slash-command help, natural-language parser, webhook-secured endpoint

Allowed tools

n8n nodes, JavaScript functions, external AI APIs, or even a small middleware microservice if n8n alone can't handle a step.

Evaluation hints

- Workflow clarity (use node groups & comments)
- Error handling & user feedback in WhatsApp
- Security of tokens and scopes
- Elegance of AI summary prompts
- Extensibility (easy to add new commands)

General Instructions for Candidates

Aspect	**Expectation**
Deadline	**10 August 2025, 23:59 IST**
Submission form	We'll share a Google Form within 48–72 h. Submit: repo URL, short description, and demo-video link (YouTube or Loom).
Partial credit	Even if you can't finish every feature, push working code and document what's left.
Collaboration	Work solo; you may use open-source libraries / AI tools but cite them in the README.
Support	If stuck, document blockers in the README rather than ghosting—communication counts.

Applicants who **deliver both tasks** (even at MVP level) will be considered first, but a polished single-task submission can still win the internship.

Good luck—build something cool!