# **AI Engineer Intern – Coding Round**

### **Objective**

Build an AI based extraction API that takes a **marksheet (image or PDF)** as input and returns a **JSON output** containing extracted fields along with **confidence scores**.  
  
**Requirements**

### **Input**

* Marksheet file (JPG/PNG/PDF).
* Max file size: 10 MB.

### **Output (JSON)**

At minimum:

* Candidate details (Name, Father/Mother’s Name, Roll No, Registration No, DOB, Exam Year, Board/University, Institution).
* Subject-wise marks (subject, max marks/credits, obtained marks/credits, grade if present).
* Overall result/grade/division.
* Issue date/place (if present).
* Each extracted field must include a confidence score (0–1).
* Json schema should be well prepared based on the type and group of the field-values

**Sample Data**

* Example mark sheets for reference: <https://drive.google.com/drive/folders/1yxsKaJbtlab8_BOFDu3bjNCoJy4zsxbX?usp=drive_link>
* These are **only for illustration**.
* Candidates are free (and encouraged) to source **additional or diverse mark sheets** (images or PDFs) from the internet or create their own samples to test robustness.
* The evaluation will be done on an **unseen in-house dataset**, so solutions must be **generalizable** (not tuned to just the sample data).

## **Tech Stack (Mandatory)**

* **Backend**: Python with FastAPI.
* **LLM**: Must use an LLM (Gemini, OpenAI, LLaMA, or other relevant open/open-source models) for structuring, normalizing, and validating extracted data.
* **Deployment**: Any platform – must be accessible via URL.
* **Repo**: GitHub (well-structured, with README and instructions).

## **Rules & Constraints**

* **Use any LLM model** (E.g: Gemini, OpenAI etc)
* **Apart from the LLM, only open-source libraries** and tools may be used (E.g: Langchain, CrewAI, Langflow etc)**.**
* Must support both **images and PDFs**.
* API should handle **multiple requests concurrently**.
* Provide meaningful error handling (invalid file, large file, wrong format).
* Keep credentials/secrets out of the repo.
* Confidence scoring must be explained (how it’s derived).

## **Evaluation Criteria**

* **Prompt and Extraction Quality (Accuracy of fields + confidence relevance)** → 60%
* **API Reliability & Project Structure** → 20%
* **Documentation (README + short approach note)** → 10%
* **Innovation / Extra Features** → 10%

## **Dos**

* Do structure your code cleanly (module wise).
* Do document setup & usage clearly in README.
* Do return a consistent JSON schema.
* Do include test samples (few mark sheets you used).

## **Don’ts**

* Don’t use proprietary/commercial APIs (Except for LLM).
* Don’t rely solely on regex/rules.
* Don’t hardcode field values (must generalize).

## **Nice-to-haves (Bonus Points)**

* API authentication (e.g., API key).
* Batch processing endpoint.
* Frontend demo page (upload + view JSON).
* Bounding boxes for extracted fields in JSON.
* Unit tests with sample mark sheets.
* Confidence calibration (explain method briefly).

## **Submission**

* **Deployed API link** (must be publicly testable).
* **GitHub repo link** with source code.
* **Approach note** (1–2 pages: Extraction approach, confidence logic, design choices etc)