```
!pip install google-generativeai fastapi uvicorn pytesseract pillow pdf2image pymupdf pyngrok
!apt-get install -y tesseract-ocr poppler-utils
```

Show hidden output

```
import os
os.environ["GEMINI_API_KEY"] = "AIzaSyDLLvgO0eViXj4Vt7C_qoa_inD6lPqSieM"
```

```
!pip uninstall -y google-generativeai
```

```
Found existing installation: google-generativeai 0.8.6
Uninstalling google-generativeai-0.8.6:
  Successfully uninstalled google-generativeai-0.8.6
```

```
!pip install -U google-generativeai==0.8.3
```

Show hidden output

```
import os
os.environ["GEMINI_API_KEY"] = "AIzaSyDLLvgO0eViXj4Vt7C_qoa_inD6lPqSieM"
```

```
import google.generativeai as genai
import os

genai.configure(api_key=os.environ["GEMINI_API_KEY"])

for m in genai.list_models():
    if "generateContent" in m.supported_generation_methods:
        print(m.name)
```

Show hidden output

```
import google.generativeai as genai
import os

# API key configure
genai.configure(api_key=os.environ["GEMINI_API_KEY"])

# Model set
model = genai.GenerativeModel("models/gemini-2.5-flash")

# Test content
response = model.generate_content("Say hello if you are working")
print(response.text)
```

```
Hello! Yes, I am. How can I help you today?
```

```
from PIL import Image
import pytesseract
import fitz  # PyMuPDF

def extract_text(file_path):
    text = ""
    if file_path.lower().endswith(".pdf"):
        pdf = fitz.open(file_path)
        for page in pdf:
            pix = page.get_pixmap()
            img = Image.frombytes("RGB", [pix.width, pix.height], pix.samples)
            text += pytesseract.image_to_string(img)
    else:  # jpg/png
        img = Image.open(file_path)
        text = pytesseract.image_to_string(img)
    return text

# Upload marksheet
from google.colab import files
uploaded = files.upload()
file_name = list(uploaded.keys())[0]

ocr_text = extract_text(file_name)
print(ocr_text[:500])  # first 500 chars check karo
```

Choose files Bhawna_Pa…me final.pdf

Bhawna_Parashar Resume final.pdf(application/pdf) - 161547 bytes, last modified: 16/10/2025 - 100% done

```
Saving Bhawna_Parashar Resume final.pdf to Bhawna_Parashar Resume final.pdf
Bhawna Parashar

parasharbhawnas {@gmalcom | +91 9958506448 | GiHub| Linkedin
Skills

Programming: Python, SQL, JavaScript

Data and ML Libraries and Frameworks: Pandas, NumPy. Sct Learn, Tensor low, Py Torch

Tools and Platforms: Power Bl, MERN Slack, Gi, GitHub, Automation Teale

CCore and Soft skills: Algocthm Design, Project Management, Problem Solving Effective communication



Work Experience
‘CNH industrial Services Pt. Lid, Gurugram June 2025 ~ August 2025 Data Analyst Intern
+ Bult Pow
```

```
prompt = f"""
You are an AI system extracting structured data from academic marksheets.

Extract:
- Candidate details (Name, Father's/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)
```

```
Return STRICT JSON only.
If any value is missing, return null.

Text:
{ocr_text}
"""

response = model.generate_content(prompt)
marksheet_json = response.text
print(marksheet_json)
```

```
```json
{
  "candidate_details": {
    "name": "Bhawna Parashar",
    "father_mother_name": null,
    "roll_no": null,
    "registration_no": null,
    "dob": null,
    "exam_year": null,
    "board_university": null,
    "institution": null
  },
  "subject_wise_marks": [],
  "overall_result": null,
  "issue_date": null,
  "issue_place": null
}
```
```

```
def confidence_score(ocr_conf=0.9, llm_conf=0.9, format_check=1.0):
    """
    ocr_conf = OCR confidence (0-1)
    llm_conf = LLM extraction confidence (0-1)
    format_check = schema check (0-1)
    """
    return round(0.5*ocr_conf + 0.3*llm_conf + 0.2*format_check, 2)


score = confidence_score()
print("Confidence:", score)
```

```
Confidence: 0.92
```

```
!ngrok authtoken "38ZYy1eR87rhndx8AoTxPqWuVKs_GRp9LD4XbSFrUfGnEiEj"
```

```
Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml
```

```
from pyngrok import ngrok
public_url = ngrok.connect(8000)
print("Public URL:", public_url)
```

```
Public URL: NgrokTunnel: "https://mignon-pseudoservile-shantell.ngrok-free.dev" -> "http://localhost:8000"
```

```
from fastapi import FastAPI, UploadFile, File
import threading
import uvicorn

app = FastAPI(
    title="Marksheet API",
    version="0.1.0"
)

@app.get("/")
def root():
    return {"message": "API is running"}

@app.post("/extract-marksheet")
async def extract_marksheet(file: UploadFile = File(...)):
    return {
        "status": "success",
        "filename": file.filename,
        "content_type": file.content_type
    }

def run():
    uvicorn.run(app, host="0.0.0.0", port=8000)

threading.Thread(target=run).start()
```

```
import threading
import uvicorn

def run():
    uvicorn.run(app, host="0.0.0.0", port=8000)

threading.Thread(target=run).start()
```

```
INFO:     Started server process [395]
```

```
from fastapi import FastAPI, UploadFile, File
import uvicorn
import threading

app = FastAPI(title="Marksheet API")

@app.get("/")
def root():
    return {"message": "API is running"}

@app.post("/extract-marksheet")
async def extract_marksheet(file: UploadFile = File(...)):
    return {
        "status": "success",
        "filename": file.filename,
        "content_type": file.content_type
    }

def run():
```

```
def run():
    uvicorn.run(app, host="0.0.0.0", port=8001)

threading.Thread(target=run).start()
```

```
INFO:     Started server process [395]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8001 (Press CTRL+C to quit)
```

```
from pyngrok import ngrok
public_url = ngrok.connect(8001)
print(public_url)
```

```
NgrokTunnel: "https://mignon-pseudoservile-shantell.ngrok-free.dev" -> "http://localhost:8001"
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.