Project Overview Automated Student Test Evaluation Agent

This project aims to automate the evaluation of student answer sheets using AI, NLP, and database automation. The system will take various correct solutions as input, evaluate students' answers based on those solutions, and update scores in a database or student portal.

---

1. Project Structure

The project consists of the following key components:

1. **Data Ingestion Module**
    - Accepts multiple correct answers (Model Solutions).
    - Accepts student answer sheets (scanned PDFs, images, or text).
    - Uses OCR (for handwritten/printed text) and direct text input.
2. **Text Processing & Normalization**
    - Preprocesses answers (spell-checking, tokenization, stemming, and lemmatization).
    - Converts handwritten answers into machine-readable text using OCR.
3. **Answer Evaluation Agent**
    - Compares student responses with model solutions using NLP & LLMs (e.g., OpenAI, Llama, GPT-4, BERT).
    - Scores answers based on similarity, keyword matching, concept coverage, and grammar.
    - Handles partial marking based on predefined rules.
4. **Grading & Feedback Module**
    - Assigns scores and generates feedback for students.
    - Uses LangChain for structured evaluation prompts to an LLM.
5. **Database & Student Portal Integration**
    - Stores evaluated scores in a database (PostgreSQL/MySQL).
    - Updates student portals via API integration.

---

2. Tech Stack

Backend:

- **Python (FastAPI/Flask)** – API to handle input and processing.
- **LangChain** – To build the AI-driven evaluation agent.
- **OCR Tools (Tesseract, EasyOCR, PaddleOCR)** – For handwritten text recognition.
- **NLP Libraries (spaCy, NLTK, Hugging Face Transformers)** – For text processing.
- **OpenAI API / Llama 2 / BERT** – For evaluating answers.

Frontend:

- **React.js (Next.js)** – Student and admin dashboard.
- **TailwindCSS/Material UI** – UI styling.

Database:

- **PostgreSQL / MySQL** – Store student marks and solutions.
- **MongoDB (optional)** – If storing unstructured evaluation data.

Automation & Deployment:

- **Celery + Redis** – Background processing (bulk evaluation).
- **Docker + Kubernetes** – Containerization & scaling.
- **AWS Lambda / EC2 / Firebase** – Cloud deployment.

---

3. Step-by-Step Development Guide

Step 1: Setup Environment

- Install dependencies:

```shell
pip install langchain openai flask fastapi tesseract-ocr
spacy nltk transformers pymongo
```

Step 2: Data Ingestion

- Create APIs to accept student answer sheets (text or images).
- Use OCR to extract text from scanned images.

Step 3: Implement LangChain for Answer Evaluation

- Define a **prompt template** for evaluation:

```python
from langchain.prompts import PromptTemplate

template = PromptTemplate(
    input_variables=["student_answer", "correct_answer"],
    template="Evaluate the student's answer:\n\nStudent's Response: {student_answer}\nCorrect Answer: {correct_answer}\nGive a score from 0-10 and feedback."
)
```

- Call OpenAI API for grading:

```python
from langchain.llms import OpenAI

llm = OpenAI(model="gpt-4", temperature=0)
score_response = llm(template.format(student_answer="Student's response here", correct_answer="Expected answer"))
print(score_response)
```

Step 4: Store Results in Database

- Save results in PostgreSQL:

```sql
CREATE TABLE student_scores (
    student_id INT PRIMARY KEY,
    exam_id INT,
    score FLOAT,
    feedback TEXT
);
```

- Python script to update DB:

```python
import psycopg2

conn = psycopg2.connect("dbname=test user=admin
password=secret")
cursor = conn.cursor()
cursor.execute("INSERT INTO student_scores (student_id,
exam_id, score, feedback) VALUES (%s, %s, %s, %s)",
               (101, 1, 8.5, "Good answer but lacks
depth."))
conn.commit()
```

Step 5: Automate Portal Update

- Use APIs to send scores to student portals:

```python
import requests

response = requests.post("https://student-
portal.com/api/update_marks", json={"student_id": 101,
"score": 8.5})
```

---

4. Advanced Features (Future Enhancements)

- **Handwriting Recognition using Deep Learning (CRNN Models).**
- **Multi-language Evaluation (Translate Answers).**
- **Bias Reduction using Explainable AI (XAI) for grading transparency.**
- **Graph-based Visualization of Student Performance Trends.**

Final Thoughts

This project will significantly reduce manpower by automating evaluations while ensuring accuracy and efficiency. Let me know if you need detailed implementation steps for any module! 🚀