PRODUCTIZETECH

AI Engineer – Computer Vision, NLP & GenAI (Full-Time Hiring Assignment)

📌 **Important Notes:**

1. **Please read the "Assignment Submission" section at the end of this document before starting the assignment.**
2. The assignment may take approximately **3–4 hours** to complete. We understand that many candidates may not finish it entirely — consider this a **fun and challenging** opportunity!
3. You are encouraged to **use any LLM** (Large Language Model) for assistance, including **generating code** to speed up development.
4. We **welcome and appreciate** the use of AI tools in this assignment.

🎥 **Watch the video "Task 1 - Explanation.mp4", "Task 2 - Explanation.mp4" & "Task 3 - Explanation.mp4" available in the shared Drive folder for a walkthrough of the tasks.**

🧠 Task 1 – RGB-Thermal Image Overlay Algorithm (15 Points)

Objective: Align thermal images with their corresponding RGB counterparts and generate overlaid outputs.

Requirements:

1. Develop a Python script that processes **all image pairs** within an input folder.
2. Each image pair follows the format:
   - `XXXX_T.JPG` – Thermal image
   - `XXXX_Z.JPG` – RGB image
3. `XXXX` denotes the shared identifier for each image pair.
4. The script should read **one pair at a time** and overlay the thermal image onto the RGB image.
5. Note: As the images are captured from two different cameras, they are not perfectly aligned by default.
6. The output should be an **adjusted thermal image** aligned with the RGB image (RGB remains unchanged).
7. Refer to the **submission folder structure** at the end of this document.
8. Sample outputs can be found in the `"sample-output"` folder.

🕵️ Task 2 – Change Detection Algorithm (10 Points)

Objective: Detect and highlight differences between “before” and “after” images of the same scene.

Requirements:

1. Write a Python script that takes an input folder containing **paired before-and-after images**.
2. The images are guaranteed to be **100% aligned**.
3. Naming convention:
   - Before: `X.jpg`
   - After: `X~2.jpg`
4. Your script should:
   - Load each pair
   - Compare the images
   - Highlight changes by drawing a **polygon, segment, or bounding box** around missing objects in the **after** image.

5. Refer to the **submission folder structure** below for output formatting.

📝 Task 3 – GLR Pipeline with Streamlit (25 Points)

Objective: Automate insurance template filling using photo reports and LLMs via a simple Streamlit interface.

Requirements:

1. Build a **Streamlit app** that accepts:
   - An insurance **template** in `.docx` format
   - **Multiple photo reports** in `.pdf` format
2. The script should:
   - Extract text from the photo reports
   - Use an LLM to interpret the template fields and detect **key-value pairs**
   - Populate the template with extracted data from the reports
3. The final output should be a **filled-in `.docx` document** based on the photo reports.
4. Use the **OpenRouter LLM APIs (Or any LLM you are comfortable with)** listed here:
   [Free LLM APIs including DeepSeek]
5. Ensure the Streamlit app:
   - Accepts the above inputs
   - Displays and allows download of the generated output

📁 Assignment Submission Structure

Create a **Google Drive folder** and ensure the following structure is followed:

➢ **Task 1:**

- `task_1_code.py` – Python code
- `task_1_output/` – Contains:

    - Input **RGB** images: `XXXX_Z.JPG`
    - Output **adjusted thermal** images: `XXXX_AT.JPG`
- **Do not include** the input thermal images (`XXXX_T.JPG`)

➢ **Task 2:**

- `task_2_code.py` – Python code
- `task_2_output/` – Contains:
    - Input **before** images: `X.jpg`
    - Output **annotated after** images: `X~3.jpg`
- **Do not include** the original after images (`X~2.jpg`)

➢ **Task 3:**

- `task_3_code.py` – Python code
- `task_3_output/` – Contains outputs for the 3 photo reports
- `task_3.mp4` – Screen recording of the **Streamlit app demo** (You can use OBS or Loom), showing:
    - End-to-end working with one photo report and the template
    - Input selection and output generation
    - **No voice-over is required**

❌ Submission Guidelines

- **DO NOT** zip the files or folders. **Zipped submissions will be disqualified.**
- Set the **Google Drive folder sharing** to **"Anyone with the link can view."**
    - Submissions without proper permissions **will not be accepted**.

Good luck! We're excited to see your creativity and technical skills in action. 🚀