Emergency Lighting Detection from Construction Blueprints

🎯 Objective:

Build an AI Vision pipeline that can extract **Emergency Lighting Fixtures** from electrical drawings and prepare **structured grouped outputs** using LLMs.

📂 Data Provided:

You will be given **10 multi-sheet electrical drawing PDFs**, each including:

- Lighting Layouts
- Legend Sheets
- Lighting Schedule Tables
- General Notes

Google Drive - https://drive.google.com/drive/folders/1d4xwvABIf_hXwGSUXTEAAACOHDl2wfyv?usp=sharing

---

✅ TASKS OVERVIEW

🔹 Emergency Lighting Detection

Train a model to:

- Detect **emergency lights**, shown as **shaded rectangular areas** on layout drawings
- Detect how many Emergency Lights are with -
  - 2' X 4' RECESSED LED LUMINAIRE

✅ TASKS OVERVIEW

🔹 Emergency Lighting Detection

Train a model to:

- Detect **emergency lights**, shown as **shaded rectangular areas** on layout drawings
- Detect how many Emergency Lights are with -
    - 2' X 4' RECESSED LED LUMINAIRE
    - WALLPACK WITH BUILT IN PHOTOCELL
    - Capture **bounding box** and **spatial location** of the fixture and nearby text/symbols

🔍 Example Output:

```
[
  {
    "symbol": "A1E",
    "bounding_box": [120, 240, 145, 265],
    "text_nearby": ["EM", "Exit", "Unswitched"],
    "source_sheet": "E2.4"
  }
]
```

🔹 Static Content Extraction

Extract static reference data to be used by the LLM:

- All **General Notes** from all sheets

🔹 Static Content Extraction

Extract static reference data to be used by the LLM:

- All **General Notes** from all sheets
- Complete **Lighting Schedule Table** including:
  - Symbol, Description, Mount, Voltage, Lumens, etc.
  - The texts extracted must be stored in the Database against the Pdf name

🔍 Example Output:

```
{
  "rulebook": [
    {
      "type": "note",
      "text": "All emergency lighting must be connected to unswitched power.",
      "source_sheet": "E0.1"
    },
    {
      "type": "table_row",
      "symbol": "A1E",
      "description": "Exit/Emergency Combo Unit",
      "mount": "Ceiling",
      "voltage": "277V",
      "lumens": "1500lm",
      "source_sheet": "Lighting Schedule - E3"
    }
  ]
}
```

◆ Define a way you will group the Lighting based on the symbols present around them?

🔍 Example Output:

```
{
  "summary": {
    "Lights01": { "count": 12, "description": "2x4 LED Emergency Fixture" },
    "Lights02": { "count": 5, "description": "Exit/Emergency Combo Unit" },
    "Lights03": { "count": 9, "description": "Wall-Mounted Emergency LED" }
  }
}
```

🧠 Best Practice Guidelines

Area	What to Encourage
Detection Quality	Use clear annotation strategies. Use shaded regions and legend symbols for training.
Text Association	Link bounding boxes with **proximal** text using spatial thresholds.
Data Cleaning	De-duplicate notes, handle rotated tables or legends, merge multi-page content.
Rule Extraction	Prefer OCR systems that preserve

🧠 Best Practice Guidelines

Area	What to Encourage
Detection Quality	Use clear annotation strategies. Use shaded regions and legend symbols for training.
Text Association	Link bounding boxes with **proximal** text using spatial thresholds.
Data Cleaning	De-duplicate notes, handle rotated tables or legends, merge multi-page content.
Rule Extraction	Prefer OCR systems that preserve layout (e.g., Donut, LayoutLM).
LLM Prompting	Provide structured context — symbols + rulebook — and request counts + classification.
LLM Evaluation	Sanity-check that output counts match raw detections and known legend rules.

📊 What We'll Monitor (Intern/Candidate Evaluation Focus)

Area	What Success Looks Like
Model Thinking	Can they design detection logic from visual cues like “shaded area”?

📊 What We'll Monitor (Intern/Candidate Evaluation Focus)

Area	What Success Looks Like
Model Thinking	Can they design detection logic from visual cues like "shaded area"?
Symbol Association	Are symbol-bounding logic and text proximity sound?
OCR Handling	Can they extract tables and free text cleanly across multiple sheets?
Reasoning with LLM	Can they structure inputs to guide LLM into reliable groupings?
Debuggability	Can they output intermediate JSONs and explain each stage's confidence?
Generalization	Can they work with multiple sheet types, rotated symbols, or low-resolution scans?

✅ API 1 – Upload and Trigger Processing

`POST /blueprints/upload`

Purpose: Upload a PDF and initiate background processing (CV + OCR + LLM)

🔶 Request:

- `file` : PDF file (multipart/form-data)

✅ API 1 – Upload and Trigger Processing

`POST /blueprints/upload`

Purpose: Upload a PDF and initiate background processing (CV + OCR + LLM)

🔶 Request:

- `file` : PDF file (multipart/form-data)
- `project_id` (optional): Project grouping identifier

✅ Response:

```
{
  "status": "uploaded",
  "pdf_name": "E2.4.pdf",
  "message": "Processing started in background."
}
```

✅ API 2 – Get Processed Result

`GET /blueprints/result`

Purpose: Retrieve the final grouped result for a given PDF name

🔶 Query Param:

`GET /blueprints/result`

Purpose: Retrieve the final grouped result for a given PDF name

🔸 Query Param:

- `pdf_name` : Name of the uploaded PDF (e.g., `E2.4.pdf`)

✅ Response (if processing complete):

```
{
  "pdf_name": "E2.4.pdf",
  "status": "complete",
  "result": {
    "A1": { "count": 12, "description": "2x4 LED Emergency Fixture" },
    "A1E": { "count": 5, "description": "Exit/Emergency Combo Unit" },
    "W": { "count": 9, "description": "Wall-Mounted Emergency LED" }
  }
}
```

🕒 Response (if still processing):

```
{
  "pdf_name": "E2.4.pdf",
  "status": "in_progress",
  "message": "Processing is still in progress. Please try again later."
}
```

📦 Submission Deliverables (All 5 Mandatory)

Send the details to - hiring@palcode.ai

Subject Linke - July_2025 | AI Vision Hands On Exercise

✅ 1. Screenshot of Annotation

- Annotated example showing:
  - Detected **emergency light** (shaded area)
  - Associated **symbols** nearby
  - Bounding boxes drawn on the electrical drawing
- Format: `.png` or `.jpg` or embedded in `README.md`

✅ 2. Hosted API (on Render)

- Deploy your API with:
  - `POST /blueprints/upload`
  - `GET /blueprints/result?pdf_name=...`
- Host on https://render.com
- Ensure background processing works and final result is queryable

---

✅ 3. Postman Collection

- Include a `.json` file or a public Postman link
- Must contain:
  - `POST /upload` with PDF file body

✅ 3. Postman Collection

- Include a `.json` file or a public Postman link
- Must contain:
  - `POST /upload` with PDF file body
  - `GET /result` with sample query
- Include example environment if needed

✅ 4. GitHub Repository

- Must include:
  - Full source code
  - `README.md` with:
    - Setup instructions
    - How background processing is handled
    - How result is stored and retrieved
  - Postman collection and sample annotated image
- Public or private repo with access granted

✅ 5. 2-Minute Demo Video

- Walkthrough of:
  - Your approach to detection and preprocessing
  - How you use LLM for grouping
  - How your APIs work and where they are hosted
- Upload to YouTube, Loom, or Google Drive (public link)