# Multi-Choice Questions Sheet Evaluator

## 🌐 Overview

This is a Python-based application to evaluate scanned multiple-choice answer sheets. It processes answer sheets in PDF format, detects student data from QR codes, identifies filled bubbles using computer vision, and compares answers with an answer key to generate a score report.

## 📄 Features

✅ Converts scanned PDF answer sheets to images.
📷 Detects and corrects sheet alignment using corner markers.
🔎 Extracts student information from QR codes.
🔹 Detects filled bubbles using Hough Circle Transform.
📚 Evaluates answers based on a predefined answers.json.
📈 Stores results in a CSV file sorted by score.
🔀 Uses multiprocessing for parallel processing.
⛔ Fully offline, no internet required.

## 📁 Folder Structure

```
project/
├── answered_sheets/
│   ├── answer_sheets_2.pdf          # Input PDF file
│   └── converted_sheets/            # Temporary PNG files per student
├── results/
│   ├── marked_<roll>.png            # Annotated evaluation images
│   └── blobs_detected_<roll>.png    # Debug: bubble detection result
├── answers.json                     # Answer key
├── results.csv                      # Final score sheet
├── main.py                          # Main evaluation script
```

## ⚖️ Answer Format

The answers.json file should look like:

```
{
  "1": "C",
  "2": "A",
  "3": "D",
  "4": "B",
  ...
  "40": "D"
}
```

## 🚆 How It Works

- **PDF Conversion**: pdf2image splits PDF into PNGs.
- **QR Extraction**: Each PNG is scanned for a QR code with roll and name.
- **Corner Detection**: Finds and aligns the sheet using L-shaped markers.
- **Bubble Detection**: Uses cv2.HoughCircles to detect answer bubbles.
- **Answer Evaluation**: Selected bubbles are compared against answers.json.
- **Scoring & Export**: Scores are saved to results.csv and annotated images.

## 🤖 Parallel Processing

Uses Python's ProcessPoolExecutor to handle multiple sheets simultaneously for faster execution.

## 📊 Output

Annotated image per student with marked answers.
results.csv showing roll, name, and score (sorted by score descending).

## 💡 Notes

- Threshold and radius values are customizable in code.
- Make sure all scanned sheets have consistent resolution (e.g., 300 DPI).
- Ensure QR codes are valid Python dict strings (e.g., {"roll": "101", "name": "Alice"}).

## 🎯 Future Improvements

- Add GUI using PyQt6 or Tkinter.
- Include negative marking or weighted questions.
- Export results in Excel or PDF format.
- Add summary charts or visualization.