# Project Documentation: SolarGuard: Intelligent Defect Detection on Solar Panels using  DeepLearning

## Project Title

**SolarGuard: Intelligent Defect Detection on Solar Panels using  DeepLearning**

---

## Project Overview and Objectives

### Extract Data

- Load solar panel images from the dataset (organized by class).

### Process Data

- Resize, normalize, and augment images.
- Prepare the data for both classification and optional object detection.

### Build Models

- Train CNN models for image classification.
- Optionally train YOLOv8 for object detection.

### Visualize & Interact

- Develop an interactive Streamlit app to:
    - Upload and classify images.
    - Display defect predictions and insights

# Technologies Used

| Component | Tools / Frameworks |
|---|---|
| Language | Python |
| Frontend | Streamlit |
| Backend/Database | MySQL |
| ML Libraries | scikit-learn, scipy, pandas |
| Visualization | Matplotlib, Seaborn, Streamlit |
| Deployment (Optional) | Streamlit Cloud / Localhost |

# Model Architecture

- Input Size: 224x224 RGB images.
- Layers: 3 convolution layers, pooling, dropout, dense softmax.
- Output Classes: Clean, Dusty, Bird-Drop, Electrical-Damage, Physical-Damage, Snow-Covered.
- Optional: YOLOv8 for object detection.

# Challenges Faced

1. **Learning Deep Learning Concepts**
    - Understanding CNN layers, parameters, and architecture design.
2. **Model Training & Tuning**
    - Ensuring balanced data and avoiding overfitting with dropout and augmentation.
3. **Streamlit UI Design**
    - Designing a responsive UI with accurate result rendering.
4. **Image Preprocessing & Normalization**

- Managing different image formats and handling grayscale or corrupted files.

5. **Model Optimization**
    - Learning and implementing pruning and quantization.

# Insights and Lessons Learned

1. **Image Data Handling**
    - How to clean and prepare image datasets.
    - Importance of resizing and normalization.
2. **Model Development**
    - Building CNNs from scratch and understanding hyperparameter tuning.
    - Comparison between different architectures.
3. **Optimization Techniques**
    - Applying TensorFlow Model Optimization Toolkit for pruning & quantization.
4. **Streamlit Integration**
    - Building a web interface from scratch and integrating with a trained model.
    - Handling file uploads and preprocessing in real-time.