

# System Specification Document

S.O.L.A.R. Project

January 24, 2026

Revision History

Version	Date	Author	Description
0.1	2025-12-17	Matteo	Initial draft
1.0	2026-01-24	Matteo	Approved release

## Contents

<b>1</b>	<b>Problem Definition</b>	<b>4</b>
1.1	Business Problem . . . . .	4
1.2	Machine Learning Problem Formulation . . . . .	4
1.3	Key Performance Indicators (KPIs) . . . . .	4
<b>2</b>	<b>Data Specification</b>	<b>4</b>
2.1	Data Source . . . . .	4
2.1.1	Plant Generation Data . . . . .	4
2.1.2	Weather Sensor Data . . . . .	4
2.1.3	Data Flow . . . . .	5
2.1.4	Data Quality and Preprocessing . . . . .	5
<b>3</b>	<b>Functional Requirements</b>	<b>5</b>
3.1	Use Cases . . . . .	5
3.2	Functional Requirement List . . . . .	5
<b>4</b>	<b>Non-Functional Requirements</b>	<b>6</b>
<b>5</b>	<b>System Architecture</b>	<b>6</b>
5.1	Training . . . . .	6
5.2	Validation . . . . .	6
5.3	Deployment . . . . .	7
5.4	Monitoring . . . . .	7
<b>6</b>	<b>Risk Analysis</b>	<b>7</b>

# 1 Problem Definition

## 1.1 Business Problem

Solar power plants are affected by strong variability due to weather conditions. Inaccurate short-term forecasts of power generation may cause grid imbalance penalties and inefficient energy dispatch. The objective of this project is to provide accurate and reliable short-term predictions of solar power output to support operational decision-making.

## 1.2 Machine Learning Problem Formulation

Given historical weather data (temperature, irradiance), predict the energy produced by the plant for the next timestamp. The dataset used provides measurements every 15 minutes.

This is formulated as a supervised regression problem using an incremental Hoeffding Tree Regressor, with an optional extension to a Long Short-Term Memory (LSTM) model for a day window forecast.

## 1.3 Key Performance Indicators (KPIs)

A custom KPI is used: for each day, the mean squared error (MSE) is calculated for each timestamp, and the daily mean is taken:

$$C_{kpi} = \frac{\sum_{i=0}^T MSE_i}{|T|}$$

# 2 Data Specification

## 2.1 Data Source

The dataset is sourced from Kaggle and represents two solar power plants. Each plant is described by two CSV files: one for power generation and one for weather conditions. This setup simulates real-time sensor data using synthetic sensors.

### 2.1.1 Plant Generation Data

Name	Description
DATE_TIME	Date and time for each observation.
PLANT_ID	Identifier of the power plant.
SOURCE_KEY	Identifier of the inverter.
DC_POWER	DC power generated at the last timestamp.
AC_POWER	AC power generated at the last timestamp.
DAILY_YIELD	Cumulative power generated for the day up to that timestamp.
TOTAL_YIELD	Total cumulative power generated up to that timestamp.

### 2.1.2 Weather Sensor Data

Name	Description
DATE_TIME	Date and time for each observation.
PLANT_ID	Identifier of the power plant.
AMBIENT_TEMPERATURE	Ambient temperature at the plant.
MODULE_TEMPERATURE	Temperature of the solar panel module.
IRRADIATION	Solar irradiation during the timestamp interval.

### 2.1.3 Data Flow

1. Raw data ingestion from CSV files.
2. Adjust missing values.
3. Standardize units.
4. Assign data to the corresponding synthetic sensor/panel.
5. Collect the data from the server.
6. Use new data points to make predictions.
7. Update the model incrementally with incoming data.

### 2.1.4 Data Quality and Preprocessing

Missing Values:

- Missing weather data are linearly interpolated.
- Missing AC/DC power values:
  - At night (no irradiation), zeros are inserted.
  - During the day, linear interpolation (with median used for better quality).
- Daily and total yield are forward-filled.

Outliers: (e.g., zero AC power during daytime) are corrected using linear interpolation.

## 3 Functional Requirements

### 3.1 Use Cases

The system shall provide the following use cases:

- Visualize the real-time power output of the plant.
- Predict the power for the next timestamp.
- Generate an end-of-day report comparing predicted and actual power.

### 3.2 Functional Requirement List

ID	Description
FR-01	The system shall ingest historical data.
FR-02	The model shall forecast the next timestamp.
FR-03	The model shall detect drift in plant power production.
FR-04	The system shall display predicted vs actual power generation per timestamp.
FR-05	The dashboard shall produce an end-of-day report for each plant.

## 4 Non-Functional Requirements

ID	Description
NFR-01	Predictions shall be generated within the 15-minute acquisition interval for near real-time decision-making.
NFR-02	The forecasting model shall achieve the custom KPI below a predefined threshold.
NFR-03	The system shall handle data from multiple synthetic sensors without performance degradation.
NFR-04	Incremental learning shall not interrupt prediction service availability.
NFR-05	The monitoring component shall detect data drift and performance degradation promptly.
NFR-06	Prediction results and measurements shall be logged for auditing and post-analysis.
NFR-07	The dashboard shall maintain at least 99% uptime during operational hours.

## 5 System Architecture

The system consists of two components: 1. Dashboard (Streamlit) for end users to visualize predictions. 2. Server-side service (Flask) for predictions and sensor data access via a DAO. The DAO provides synthetic sensor readings to simulate a real-world scenario.

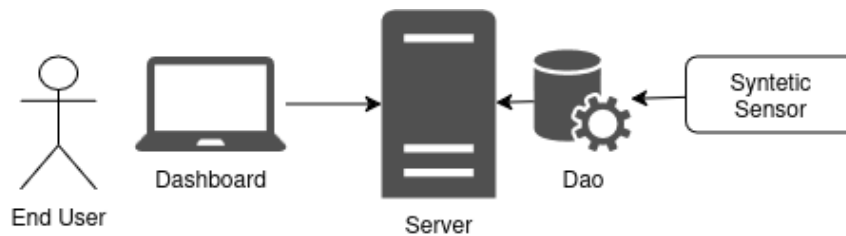


Figure 1: High-Level System Architecture for Solar Power Forecasting

### 5.1 Training

- The model is trained incrementally from the first data point.
- Each new sensor reading immediately updates the Hoeffding Tree Regressor.
- No offline batch retraining is required; the pipeline supports fully online learning.

### 5.2 Validation

- Online rolling evaluation monitors the custom KPI (daily mean MSE).
- ADWIN monitors data streams for drift in real time.
- Alerts are triggered if drift is detected, allowing operators to adjust updates.

### 5.3 Deployment

- The incremental model is exposed via a Flask API.
- The Streamlit dashboard requests predictions from the Flask service.
- The DAO fetches the latest sensor readings for the model.
- Predictions are returned to the dashboard for near real-time visualization.

### 5.4 Monitoring

- ADWIN monitors feature distributions and prediction errors.
- Metrics and predictions are logged for auditing.
- Alerts are triggered on significant drift detection.

#### Process Flow:

1. User requests a prediction via the dashboard.
2. Flask queries the DAO for latest sensor readings.
3. Sensor data is input to the incremental model.
4. Prediction is returned to the dashboard.
5. The model updates incrementally with each new data point.
6. ADWIN monitors for drift and triggers alerts if necessary.

## 6 Risk Analysis

Since preprocessing is defined and validated at initialization, runtime data transformation risks are minimized. Remaining risks concern data distribution changes, model behavior, and system operation.

Risk	Description and Mitigation Strategy	Impact
Data Drift	<b>Description:</b> Statistical properties of sensor data may change over time due to seasonal or environmental variations. <b>Mitigation Strategy:</b> ADWIN monitors the data stream in real time and triggers alerts when drift is detected.	High
Concept Drift	<b>Description:</b> The relationship between weather features and power output may evolve, reducing prediction accuracy. <b>Mitigation Strategy:</b> Incremental Hoeffding Tree continuously adapts to changes in the input-output relationship.	High
Prediction Degradation	<b>Description:</b> Model performance may decline gradually despite incremental updates. <b>Mitigation Strategy:</b> Continuous KPI monitoring (daily mean MSE) with alert thresholds.	Medium
System Availability	<b>Description:</b> Service interruptions may prevent users from obtaining predictions. <b>Mitigation Strategy:</b> Lightweight, modular Flask service reduces failure points.	Medium

Monitoring Limitations	<p><b>Description:</b> Poorly configured thresholds may fail to detect drift or degradation.</p> <p><b>Mitigation Strategy:</b> Combined use of ADWIN and KPI monitoring ensures reliable detection.</p>	Medium
---------------------------	--	--------