# Renewable Energy Plant System (REPS)

## General Overview

A Renewable Energy Plant (REP) is a multigeneration energy system plant that manages the production of renewable energy for cities, industries and so on.

Renewable energy sources are usually accessible at any location in the world, depending on the type. Also, using renewable energy resources has several advantages and benefits compared to conventional energy sources.

The REP is equipped with software that controls everything in the plant, e.g., from energy production to site surveillance. In order to do that, it collects information from sensors and cameras. Sensors collect, elaborate, and send to a server control room data about the health of the production system, such as the production of energy/hour, while cameras register and send videos to the Server staff room of what is happening in the plant, such as the circulation of people and transportation means (for surveillance activity) or the exchange of stuff by truck loading.

Not necessarily everything has to be sent to the server and remotely processed. A local subsystem (e.g. embedded processing in cameras) can be considered, for example, as based on a framework for embedded image processing.

The REP software is also aimed at improving energy production by, for example, moving the solar panels to accurately follow the sun, or moving the wind turbines to follow the wind direction. Also, REP can suggest changing production stuff for any reason, e.g., due to less energy production for ageing.

## Implementation Goals

Your group has been hired by a renewable energy company to design and build a renewable energy plant system. The goal of the project is to create a functional and efficient power plant that harnesses renewable energy sources to generate electricity for local communities. The power plant must be reliable, sustainable, and cost-effective. You are required to implement the following use cases:

1. The system should be designed to monitor and control the power plant's renewable energy sources, including solar panels, wind turbines, and hydropower.
2. The system should be capable of collecting data related to the energy generated by renewable sources and storing it in a file.
3. The system should provide a view of the power plant's energy generation and storage capacity, allowing operators to adjust the power plant's operation as needed. This view should show the data stored in the file.
4. The system should be able to analyse the data collected from renewable sources. It should be possible to filter the data on an hourly, daily, weekly and monthly basis. It should be possible to sort the data where possible. It must allow a user to search for required data stored in the system.
   Data Analysis should include:
   - **Mean**: the average, which is found by adding up all the values in a set of data and dividing it by the total number of values you added together.
   - **Median:** the middle number in the set of values. You find it by putting the numbers in order from the smallest to largest and covering up one number on each end until you get to the middle.
   - **Mode:** the number or value most often in the set. To find the mode, you need to count how many times each value appears.
   - **Range:** the difference between the lowest and the highest value. To work it out, simply subtract the lowest value from the highest.
   - **Midrange:** the number that is exactly halfway between the minimum and maximum numbers in a set of data. To work out the midrange, you must find the sum of both the smallest and largest, and divide it by 2.
5. The system should be able to detect and handle issues with renewable energy sources, such as low energy output, and equipment malfunction, generating alerts for the operators accordingly.

**Note:**
Make sure to follow the Scala style guide and use appropriate naming conventions for variables, functions, and classes. Your code should be well-organized, well-documented, and easy to read. All the functionality and code structure **must** follow the Functional Programming paradigm where:

- Data immutability is ensured.
- Iterative works are done with the help of recursion.
- Higher-order functions are used to accomplish tasks.

- Error handling in a functional way. Do this as much as you can. We don’t expect perfectionism at this point.
- Demonstrate the other learned concepts as much as you can, e.g., type parameterization, currying, etc.

Implement the file I/O in a non-functional way in Scala – the usual way of imperative languages. Better to do I/O either with .xls or .csv files; it is relatively easier.