**Functional design Plant Patrol**



**Luuk Vlug**

**500807974**

[**luuk.vlug@hva.nl**](mailto:luuk.vlug@hva.nl)

**Functional Design Plant Patrol**

**Index**

[**Overview of the webpage**](#_rt4wrg41ug6) **3**

[**Sensor result**](#_ut2audszw11v) **4**

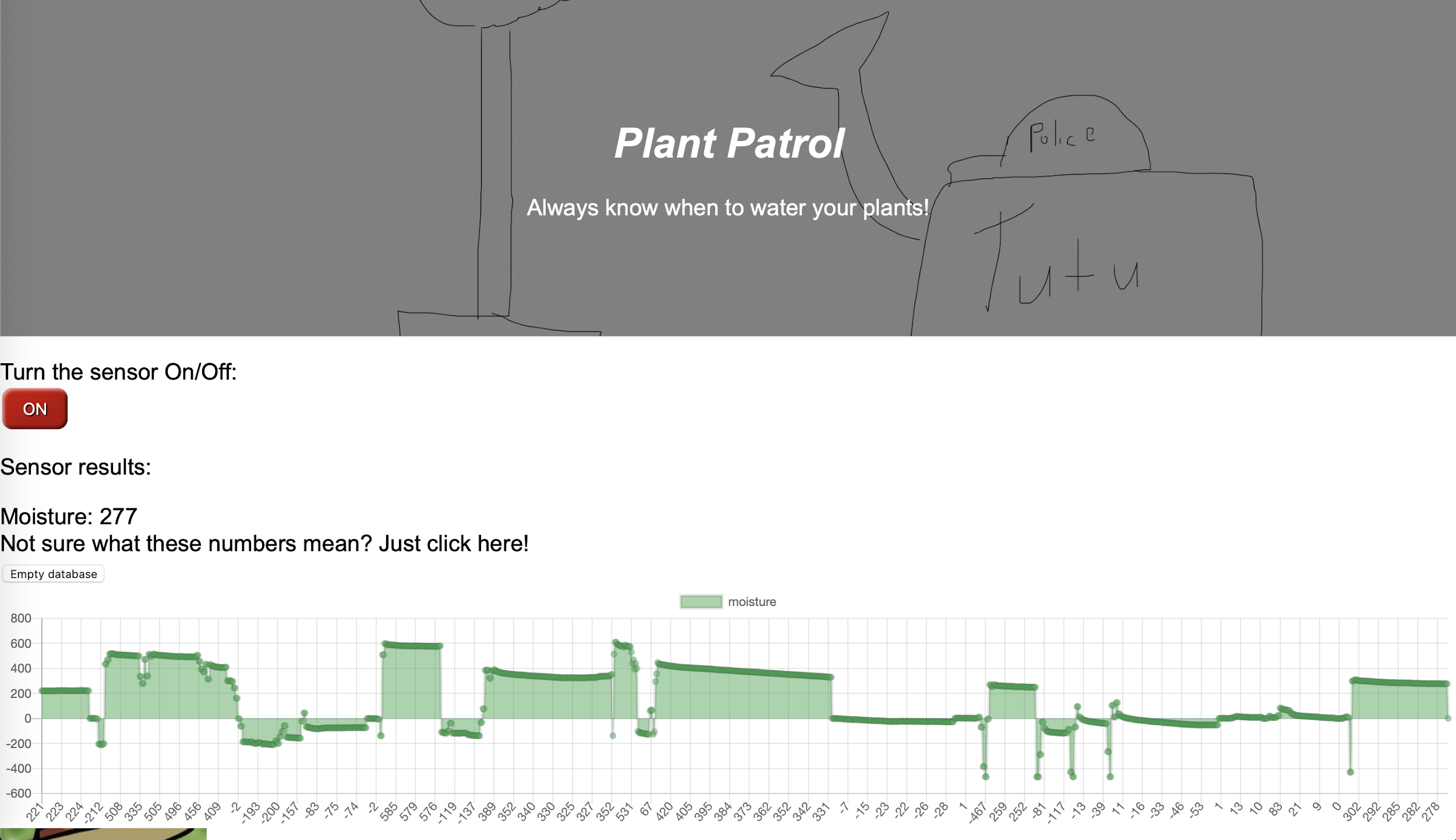
[**The graph with sensor results**](#_wf24p9zeu8j4) **5**

[**Status of the plant**](#_7ogem1xplpet) **6**

[**Forecast API**](#_ktkpx2g7x189) **6**

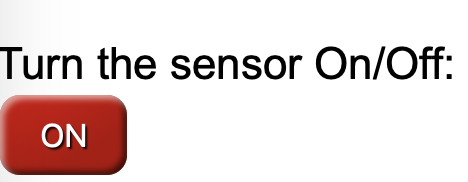
# Overview of the webpage

When opening the webpage you are greeted with this sight:



**Sensor control from the website**

In this screenshot you can see an image and the name of the website and the slogan. Below that you will find a button:

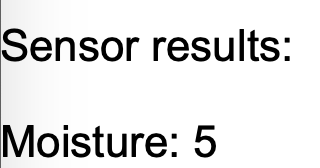


With this button you will be able to turn the sensor on and off. The current status of the sensor is displayed on the button, in this case ‘ON’, and shown with a LED on the D1 Mini.



# Sensor result

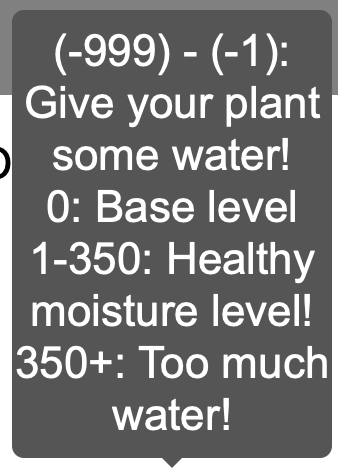
The next thing you will see on the page is the sensor result.



These results will keep updating as long as the sensor is connected and turned on. When looking at this value you might be confused as to what it means. Luckily below the moisture value, you will see this sentence:



By clicking on this sentence, a useful popup will appear, giving you information about the values on your screen. This popup looks like this:



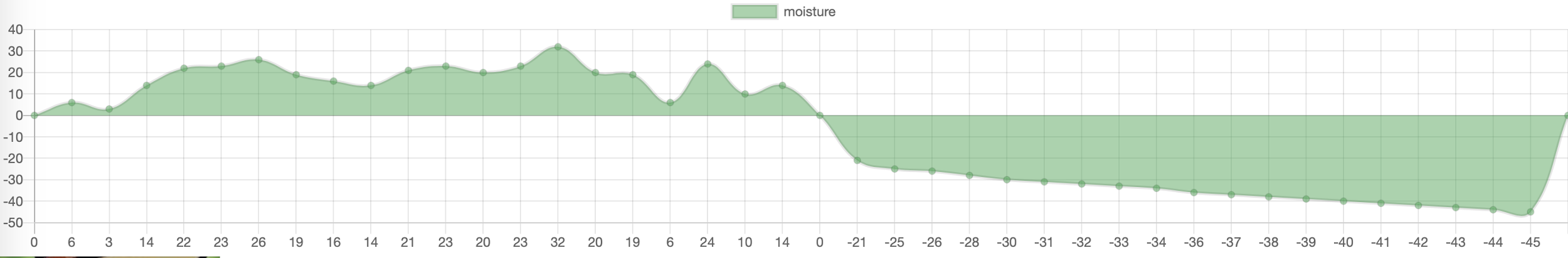
The moisture values are split up in categories, to give users a clear overview so they can know in what category their plant currently is. With this information users have an easier time deciding whether the plant needs water, has too much water or is just fine.

# The graph with sensor results

Below the written sensor result, you will find a button. This button is called ‘Clear graph’.



Everytime a new moisture result is picked up by the sensor, it is both displayed on the website, in the form of ‘Moisture: xx’ and the value is added to the graph. However I don’t want to restrict users to only being able to use the sensor on just 1 plant. Therefore I want users to be able to clear the graph before sticking the sensor into another plant.

The graph where all these moisture values are added into looks like this: 

As you can see in the picture, the graph can both display positive and negative values. Because sometimes the sensor can fluctuate a bit, trying to find the definitive value, this graph is a useful tool. If for example the sensor is still fluctuating, you don’t have to constantly look at the moisture value to see if it’s displaying equal values. You can instead just look at the graph and see if the values are still longer dropping.

# Status of the plant

Instead of just having the moisture sensor result and the graph to display the current state the plant is is, I also added 2 pictures. These pictures are displayed on the same spot, just are a version of a happy and a sad plant. There is a maximum and minimum the moisture value should be in to make the happy picture appear, if outside of this range, the plant will turn sad. This indicates the plant has too much, or too little water. These images look like this:



The range for a happy plant is: -35 - 350, anything outside of this range will make the sad plant appear.

# Forecast API

Because some people also have outside plants, I decide to base my API around the idea that it is sometimes hard to decide whether you should give your outside plants water. If you give them water, and a few hours later it’s starting to rain, the plants might very well drown. If you expect it to rain, but it ends up not raining, the plants might die from drought. To help people make the decisions to water their outside plants or not, there is a useful forecast API.

Using the textbox, you can enter a city (not limited to the Netherlands), and by pressing submit. This textbox and the button look like this:



After submitting the city, the API will look up the forecast for this specific city. The information it will display is the temperature, a description of the weather and of course a time and date for the forecast. For ease of reading, I decided to display this information in a table. Here you will see how this table looks when filled data (in this case from Amsterdam):

