A00279259

NATALIA PALEJ

*Software Design with Artificial Intelligence in Cloud Computing*

# Introduction

Our goal in this project is to create weather station using Python Language and then running it on HTML by using Flask.

# Specs

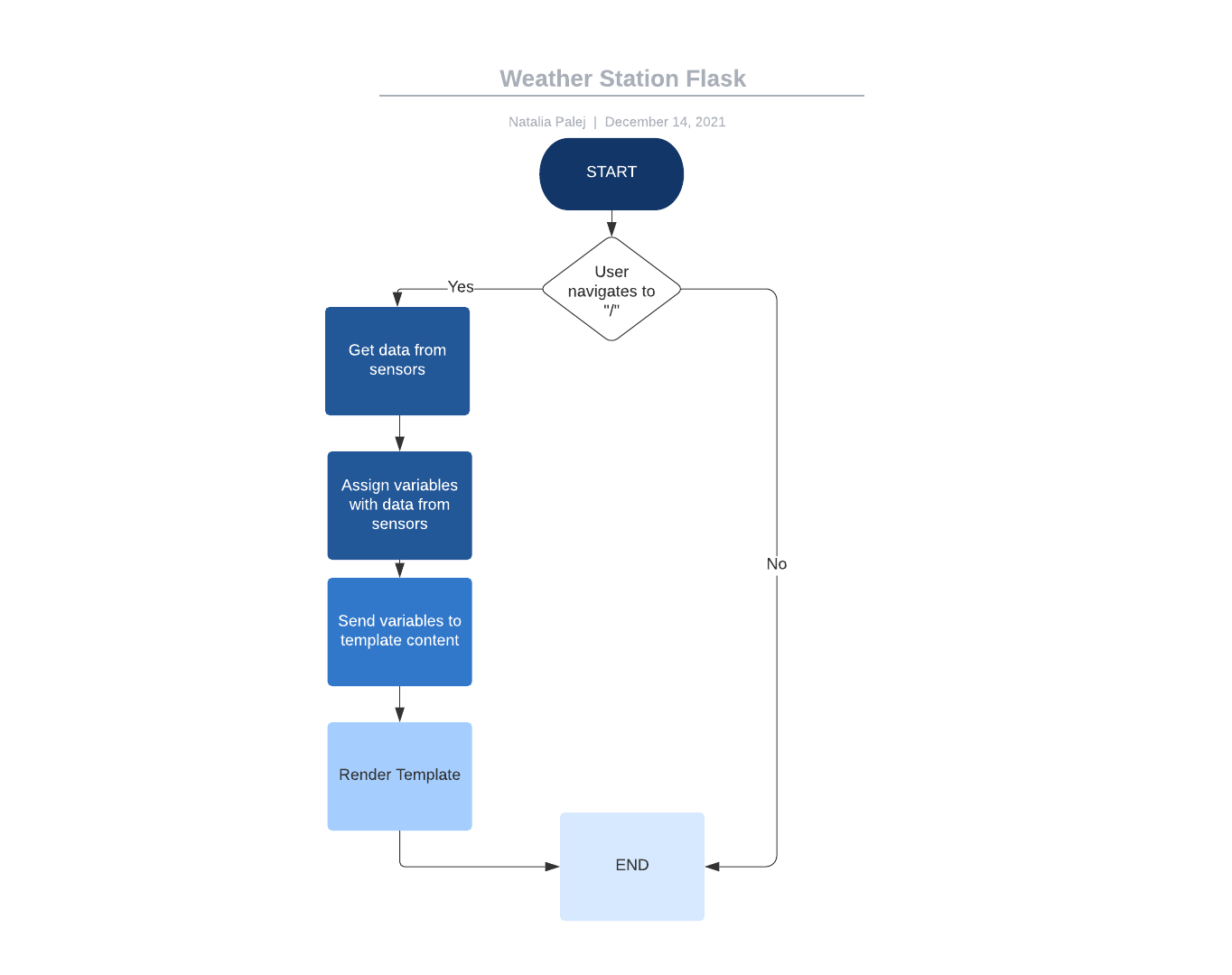
# Design

## Hardware

Raspberry Pi is an extraordinary device, it is known as a single board computer which means exactly what it sounds like: it is a computer just like a desktop but built on a single printed circuit board. Raspberry Pi is very small - roughly about the size of a credit card.

…

To further develop and test our application outside of college hours, we have used Trinket. Trinket is a page where you can write and run code in any browser on any device. Trinket introduced Sensors Emulator which we took advantage of to test the code changes when we did not have access to Raspberry Pi device.



## Software

Python is a scripting high-level programming language. Python supports libraries and packages, which encourages program flexibility and code reuse.

# Build

## Hardware/Software

At the beginning of our Python program, we have imported all necessary libraries (sense\_hat, flask) and created an instance of SenseHat object. Afterwards we initialized a flask app.

Graphical user interface, text

Description automatically generated

Once we had that done, we created our first and only app route mapping “ / “. We only need one @app.route as we are going to use one html lading page to display data from the sensors. In the index() method we created SenseHat object and used it to fetch data from available sensors, then we assign them to variables called “celsius”, “fahrenheit”, “humidity”, “pressure”.

Text

Description automatically generated

Finally, the last thing we had to do inside the index() method was to return the result to the end user. To do that we took advantage of render\_template method from Flask. As seen in the screenshot below we are sending previously assigned variables to the context of the html page. Render template method takes one or many arguments and it renders a template from the template folder with the given context:

Parameters:

**template\_name\_or\_list** (Union[str, List[str]]) – *the name of the template to be rendered, or an iterable with template names the first one existing will be rendered*

**context (Any)** – *the variables that should be available in the context of the template*.

Text

Description automatically generated

Lastly, to expose our application to all devices on the network, we must add the following code. Here the ‘host=0.0.0.0’ means the web app will be accessible to any device on the network. We also enable the debug mode. Flask Debug mode allows developers to locate any possible error and as well the location of the error, by logging a traceback of the issue.

Text

Description automatically generated

Once we run our app, we can see the following:

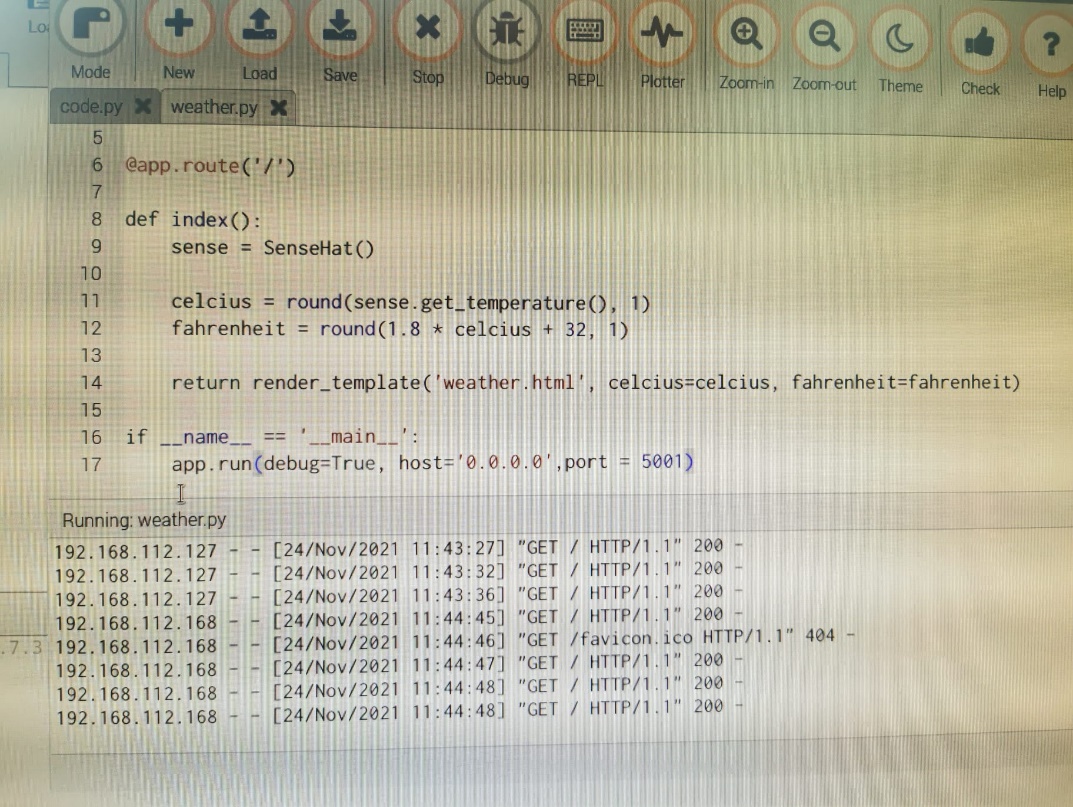
Text

Description automatically generated

We can see our Flask application is up and running on <http://192.168.1.11:5000/>. Additionaly, we can see the details about the debugger which we set to be enabled. Details like debugger PIN are crucial if we ever want to execute arbitrary Python code from the browser for debugging purposes.

# Test

We have managed to start the project during our labs. Below are two pictures that represent working code and proper values from the actual Raspberry Pi sensor.



# Text Description automatically generated

We have accessed the app from two different devices hence why there are two different IP’s that are getting the returned template. As we continiued with our weather project, unfortunately we did not have access to Raspberry Pi and decided to use Trinket emulator for testing purposes. As we wanted to read humidity and pressure as well, we ran the code in the Trinket page to make sure the code works properly. We chose random sensor values and got expected result.

Graphical user interface, text, application, chat or text message

Description automatically generated

Once we got the values, we wanted to get them displayed in our improved html page. We managed to achieve that by commenting out actual code that was reading Raspberry’s sensors and replaced it with hard-coded values that we got from Trinket to simulate the real values scenario in our application without the use of Raspberry Pi.

Text

Description automatically generated

Graphical user interface, application, website

Description automatically generated

# Conclusion

At the start, making this project was quite demanding as I have never used Flask in the past. It took me some time to understand how excatly it works, and how the app reads the context. Once I understood the idea of Flask, the project progress went much smoother and quicker. I feel like I learned a lot from it, and it is something I would love to do more in future. The only downsight was the lack of actual Raspberry Pi device, as it would be even more fun to play with. We could create additional decisions in our code, for example, to light up LED display in specific colour once the temperature reaches certain value. Each time I work with Python, I am more and more amazed how powerful but also simple it is.

Contents

[Introduction 1](#_Toc90416856)

[Specs 1](#_Toc90416857)

[Design 1](#_Toc90416858)

[Hardware 1](#_Toc90416859)

[Software 3](#_Toc90416860)

[Build 3](#_Toc90416861)

[Hardware/Software 3](#_Toc90416862)

[Test 5](#_Toc90416863)

[Conclusion 8](#_Toc90416865)