# ENSF 310

## Lab 4: Software Design Process

Edit this file and write your solutions to the problems in the empty markdown blocks. Once you are done, check your lab into your github repository for full marks.

# Software Design

## Problem Statement

You have been asked to develop a simulator for the [Nanoleaf light panels (https://nanoleaf.me/en-CA/products/nanoleaf-light-panels/)](https://nanoleaf.me/en-CA/products/nanoleaf-light-panels/) installation outside of the the Zetta space on the 2nd floor of ICT. This simulator will be used to test possible interactive displays. A photo of the installation is below.



[An example of another extensive installation can be found here (https://inhabitat.com/nanoleafs-terra-light-wall-paints-arctic-landscapes- with-1200-color-changing-leds/).](https://inhabitat.com/nanoleafs-terra-light-wall-paints-arctic-landscapes-with-1200-color-changing-leds/)

**NOTE**: You do not have to write any code as part of this lab!

### Inception Phase

Spend some time brainstorming alternatives which you may consider and whether you think this simulator is technically feasibly. Things to consider include programming language, libraries, existing toolkits, existing software, etc.In the space below, describe any alternatives and the technical/economic feasiblity. You must discuss at least three alternatives.

In the space below, describe any alternatives and the technical/economic feasiblity. You must discuss at least three alternatives.

**Inception Phase**

Looking at it firsthand the simulation would not be as technically feasible as some of the alternatives given below but is still a good way to test out and serves as a good simulator for the Nanoleaf panels.

**Alternative-1**

The input itself can have a few alternatives i.e., how the simulation is going to get the input. There can be many ways, but touch and sound seem interesting as it gives much more interactive ability. With these two styles of input there can be other features such as mini games which can be designed into it.

**Alternative-2**

The second alternative could be due to different programming languages for example there is a **JAVA** and **Python** both are suitable for the simulation we are going for. Java is a statically typed and compiled language, and python is a dynamically typed and interpreted language. This single difference makes Java faster at runtime and easier to debug, but python is easier to use and easier to read.

**Alternative-3**

We can also investigate APIs for example REST APIs support more feature than HTTP APIs, while HTTP APIs are designed with minimal features so that they can be offered at a lower price and would be more economically feasible. However, REST will offer more features which will be needed for the simulation.

Overall, the simulation would be economically feasible as Python would be used and some readily available libraries and other toolkits which are free/low cost. Looking at the technical side, it should be feasible as the tools/libraries which are going to be used will be giving us enough technical freedom to implement and modify the simulation to our needs.

### Requirement Analysis Phase

In the space below, analyze the requirements. State which ones are given by the problem statement, and which are assumptions. Your requirements must be **testable**.

**Requirement Phase**

There are a few requirements already given in the problem statement so we will look at them first.

1) The software to be designed is a simulation for the Nanoleaf Light Panels.

2) The simulation should be interactive i.e., there should be one or more input options and some number of outputs.

3) The simulation should be able to display all colors in any type of order or orientation (this can be seen in the pictures).

4) There should be options for personal customization regarding both input and output.

5) Nanoleaf should accept any valid RGB value.

Finally, we can have some assumed requirements that would be such as, the simulation should be fast enough, efficient, and the type of input for example through touch, voice etc.

### Design Phase

In the space below, describe the software design and architecture. You may use Jupyter markdown to create tables to organize your thoughts. Your design should include the input and outputs of any functions.

**Design Phase**

The following are some of the important functions

1. Switch – A function to turn the panels on/off.
2. Triangles – A function to create a 2-d array of triangles
3. Input – A function to receive input. (Can receive both types of input, touch/sound)
4. Validity - A function to check if the RGB values are under 255.
5. Transmit- A function to output these colors onto the triangles.
6. Pattern- A function to select or customize the pattern of lights.
7. Cooldown – A function which controls the time taken to switch colors.

### Construction Phase

What information is necessary to do the actual software development? **Note**: you do not have to code your design.

**Construction Phase**

The designing team should have someone (if not everyone) who knows how the Nanoelaf Technology works.

Similarly, someone should know and be adept with python and or Java.

Moreover, basic knowledge about some simple games and how colors work could help in designing inbuilt patterns for example purple to yellow.

### Testing Phase

How would you test your implementation?

**Testing Phase**

Each function ranging from the input to the output will be tested.

There is a function included in the design phase as well which check if the input is valid i.e., RGB values are under 255.

Every function can be tested to the respective limits and if it exceeds or is outside the range a message will be displayed.

1. **Maintenance**

What maintenance would be necessary?

**MAINTENANCE**

This design is subject to additions and changes as there is a pattern function which can be updated to add more and more functions thus it is configurable and flexible to your imagination. Apart from this It can be modified to cope up with changing requirements. Due to its simple design the design can be easily read and edited to deal with errors which may arise due to unexpected situations.