**Beet Seed**

1. For this exercise, I chose a **toaster** to be my test object.

* Atomic: this criteria means testing one single functionality of the product individually.

Requirements:

✔️ The toaster needs to have a cancel button to stop the toasting process at any time the user wants to;

✔️ The toaster needs to have a functionality to raise and lower the toast inside of the toaster (lower to toast, raise to take the toast off);

✔️ The toaster needs to have a button to change the temperature.

✔️ The toaster needs to have a tray that comes out easily and to where the crumbs can go so it facilitates the cleaning afterwards.

* Consistent: the features of the product must be as consistent as possible (if it behaves in a way today, it should behaves the same way tomorrow)

Requirements:

✔️ The toaster should consistently/always produce a sound when the toasting is complete;

✔️ The toaster should consistently/always lift the toasts when they’re ready;

✔️ The toaster should consistently/always toast the bread the same way on both sides.

* Testable: a testable requirement is one that is possible to validate and verify through tests to the product that is being tested.

Requirements:

✔️ We can try to prove (test) if there’s indeed a sound when the toasting is complete;

✔️ If the toaster requires, for instances, 1 minute to toast the bread (or if we selected specifically that time), the toaster should lift the already toasted bread at that time (we can also test this);

✔️ We can also test the *security* of the toaster (this is a non-functional test, I believe): if the bread gets stuck and it doesn’t pop up when the time is ready, it would be great if the toaster would turn itself off, so the bread doesn’t keep toasting and eventually burn.

* Traceable: means that the tested product is well documentable and identifiable.

Requirements:  
(*I had some difficulties on this one because I don’t know if I understood correctly how to apply to the toaster example*)

✔️ Assuming that the toaster has an instructions manual, if the toaster takes 1 minute, for instances, to toast the bread, that should be well documented on the instructions manual;  
✔️ If the toaster has an additional tray to the leftover crumbs, the manual should have that written and also explain how to take it off and what are the best ways to clean it;

✔️ The instructions manual must describe all the important instructions about how to make the toaster work.

**Beet Sprout**

1. ✅
2. I believe that the best to achieve the highest possible quality of the final result would be combine all the techniques.

But, if I had to choose just one, I’d probability go for the review technique.

It’s important to have different opinions and ideas about everything. This can also be applied to the requirements.

It’s good to have other opinions about the requirements, to see if they are clear or if it misses something. The goal here is to see, right at the beginning of the process, if the requirements are good to be applied because if there’s any mistakes, it’s better to solve them at the start, rather than later and taking the risk of missing something important.

(So, we can do a superficial overview, a technical inspection and a formal inspection.

On the superficial overview, it’s like a more general review of the requirements, on which our colleagues can give us suggestions, ask questions about the unclear requirements, add comments.

On the technical overview, we invite professionals of that specific area (of the product to be tested) to give their opinion about the requirements. The idea here is to have them telling us if, in their professional point of view, something’s missing, what we can add, if everything is clear and making sense.

Then, we also have the formal inspection. Here, we have a larger number of experts on that matter and, in a more serious approach, they review the requirements in a stricter way. It involves a lot of documentation and it takes a lot of time.)

**Mighty Beet**

1. ✅
2. Functional requirements:

* The users should be able to create an account and log in effectively;
* The users should be able to upload photos of their cats;
* The users should be able to report negative comments and/or against the rules if they need to;
* The users should be able to allow (or disallow) other users to see their posted pictures (for instances, disable that a specific user sees their pictures, or a specific picture, or whatever they prefer;
* The users should be able to like other users pictures and/or follow them;
* The users should be able to send messages to other users, and receive too;
* The users should be able to search for specific content, like a specific cat breed, for instances.

Non-functional requirements:

* The app should load the content quickly, even if there’s many other users using it at the same time;
* It should be secure for users to use the app and keep their personal information there;
* The app should function on every operating system and/or web browsers;
* The app should be inclusive for everyone, like blind users *(I don’t really know if this would work as it’s an app for photos, but maybe somehow a verbal description of the photos?*) and/or photosensitive users (for users who have epilepsy, for instances, it shouldn’t have flashes and strong lights and colors);
* The users should be able to change the language of the app (translate it) and the translations should be correct and accurate.