Final Project Report

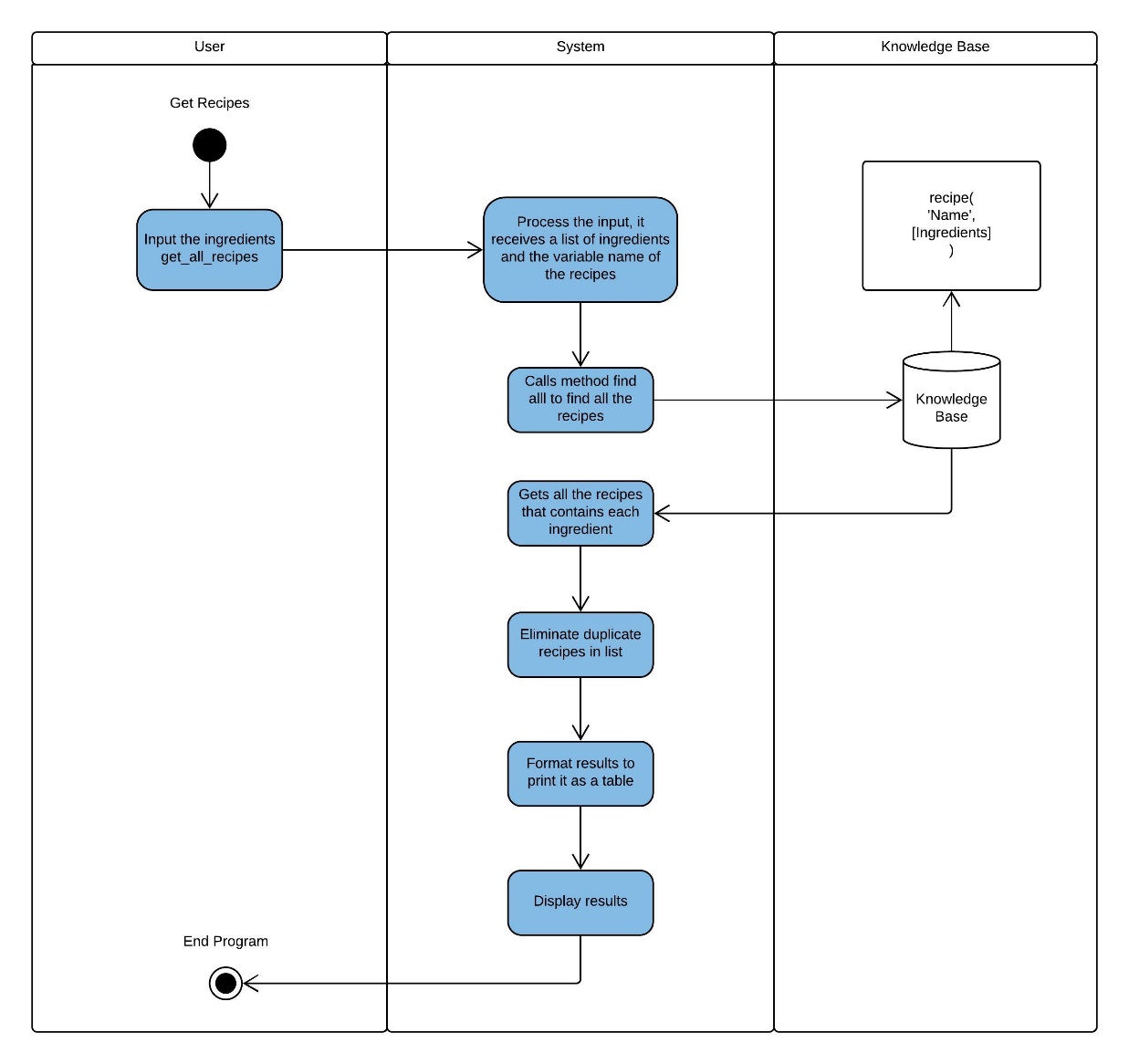
**Context of the Problem:**

When you start living by your own it’s hard to know what kind of food you can prepare and what are the ingredients for each type of food. You have to search in the internet or even ask somebody to know the recipe just to realize you don’t have those ingredients. And even when you do have all the ingredients needed, you don’t know how to do it because you have no knowledge of the recipe.

**Solution**

The solution I created is a program that will help the user in different scenarios, like for just knowing recipes or as a tool to use in real life cases where you’re hungry and you want to do something to eat with all the ingredients you have in your house but there is not enough time to look it up on the internet, or there isn´t even an internet access.

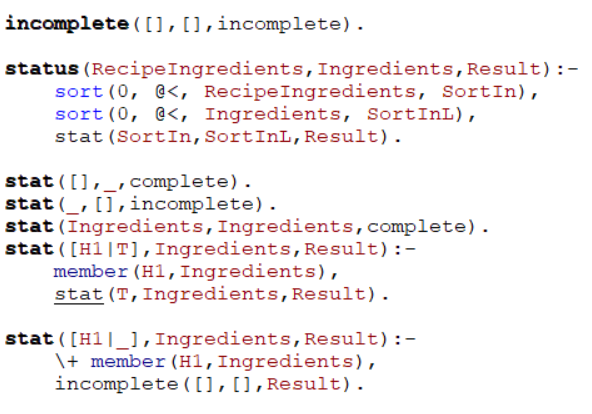
The user will be able to type the ingredients that he/she has and a set of results will appear base on the coincidences found. These results will be presented in a table where it will contain the ingredients and the name of the recipe, the status of that recipe so when status is complete it means you do have all the ingredients needed for that recipe and when it is incomplete you need more ingredients.

The program works as follows:

**Why Prolog?**

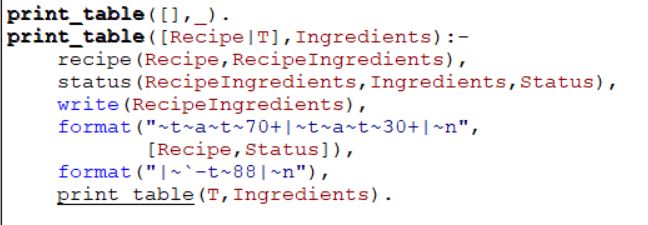
It seemed like an intuitive approach to use prolog because of the complex of the problem, once I had defined my knowledge base it’s easier to search for results in an appropriate and efficient way and defining the searching algorithm was too. With the advantage of using backtracking and a set of recursive rules, the piece of code in charge of the search was really short. Regardless, this doesn’t mean it was an easy task; there were different challenges I had to face which I’m going to describe:

1. Define the status rule which defines if you have all the ingredients to make the recipes given by the findall function. The code works as follows:



First, I had to order the list of ingredients per recipe given by the findall function in an alphabetic way, order the list of ingredients given by the user too. Then it compares if the HEAD of the recipe ingredients exists in the ingredients given by the user, if it exists continues the recursion with the tail but if it doesn’t exist it stops the recursion and set it as incomplete.

1. Print table rule gave me a lot of trouble too because I had to research how to use the format/2 function of prolog as shown here:

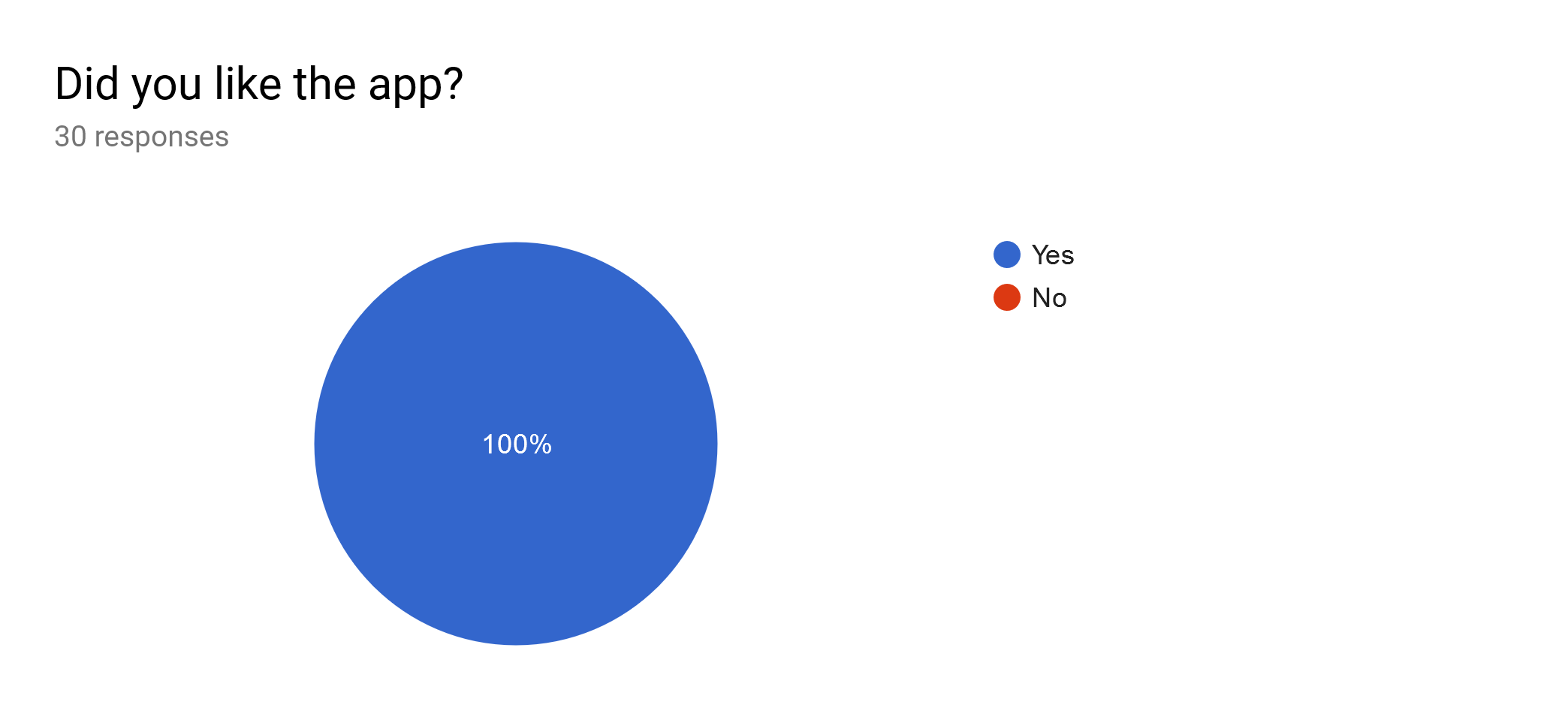


**Results**

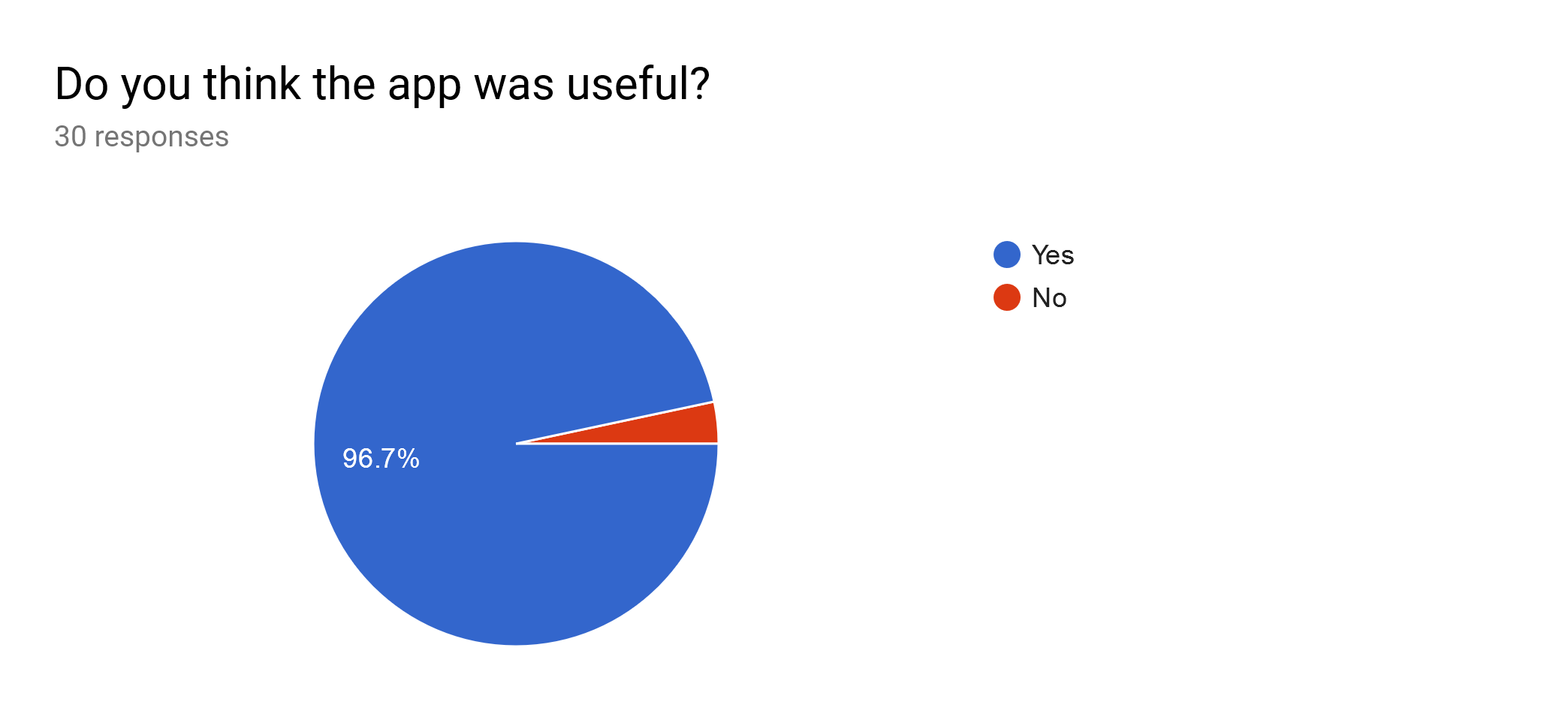
I showed the program to 30 users and make test of it just by typing some ingredients, thanks to this I could add more recipes that I hadn’t think about and the program worked as expected every time.

One of the problems I found was that the people that don’t know prolog were a little bit confused on how to use the program but once I explained them how the input format was, they could do some tests with the program.

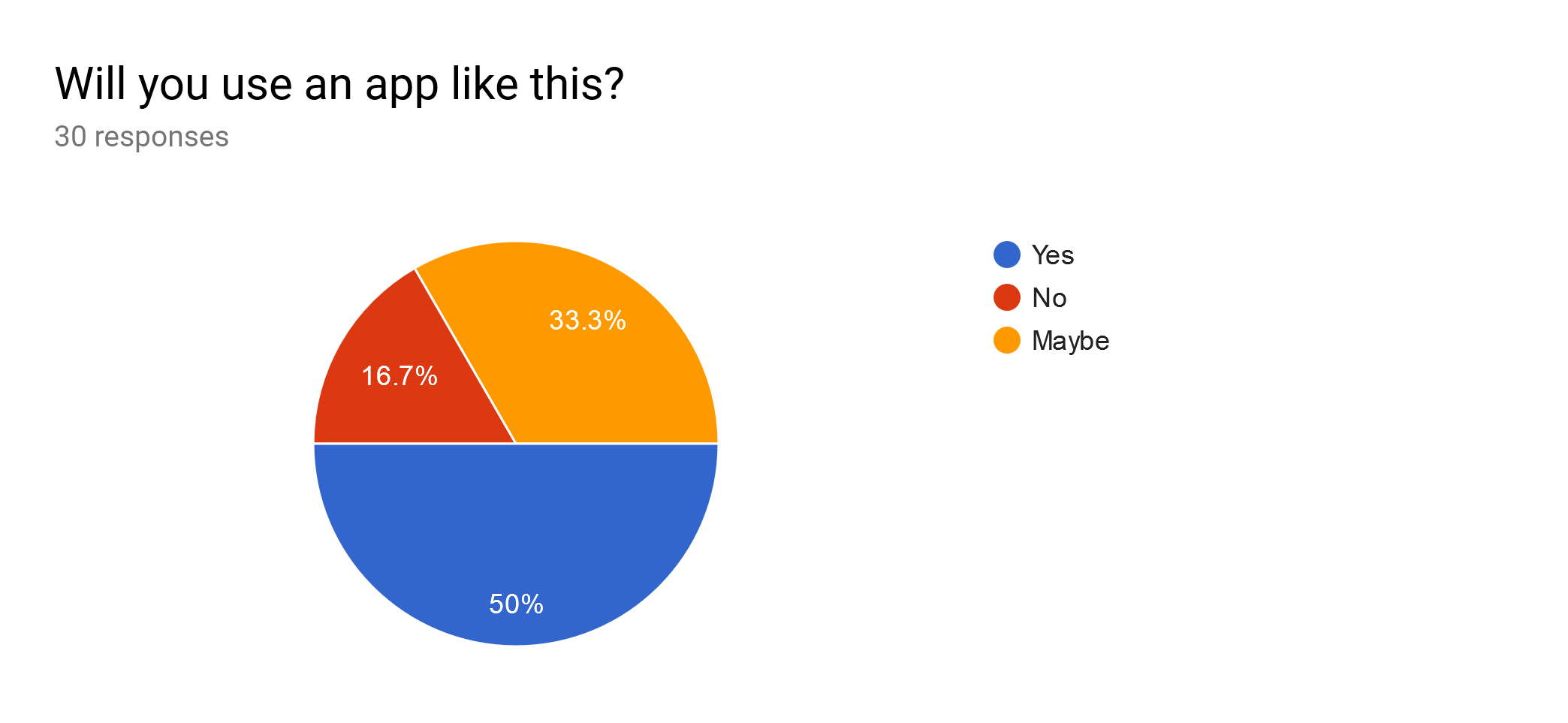
After the user tested the program I asked him/her to fill a google form and the results were as follows:



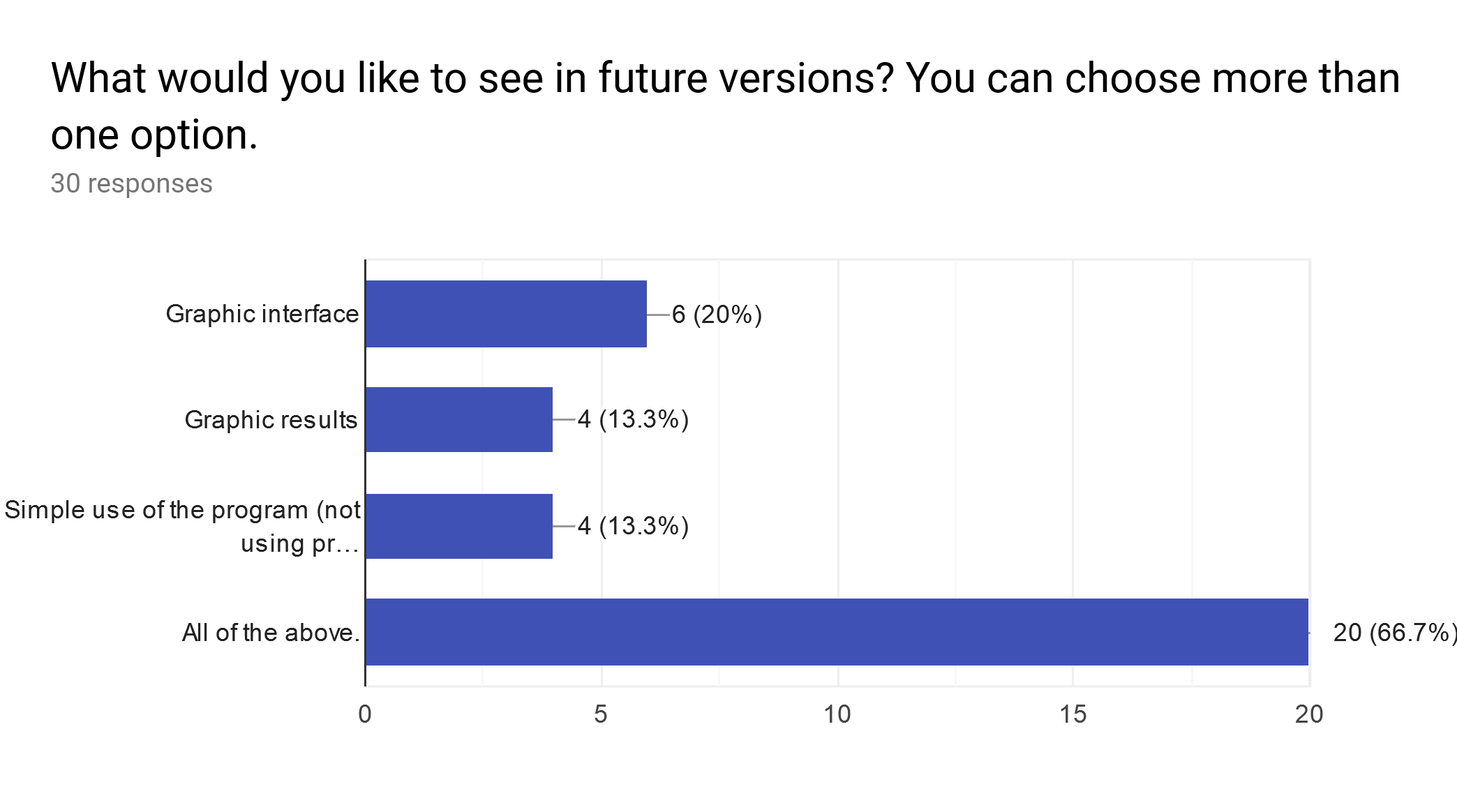
All the 30 persons liked the app even when they had some comments about it like the user interface wasn’t really understandable for all of them.



Of all the 30 persons just one of them answered that he/she didn’t think the app was useful.



Here is a very interesting result, even when almost all of them answered that they liked the application and they think the app was useful, not all of them will use an app like this, 33.3% of them answered as a maybe and one of the reasons I head the most was the lack of time to do this.



The last question was about future versions of the program, a 66.7% answered that they would like to see a graphic interface with results in a graphic way and a simple use of the program like just typing the ingredients or even with a checklist instead of using Prolog format.

**Conclusions**

In the end, what I got was a nicely working program with real data about the digestive system that will help make the patient’s care more efficient and decrease the margin of error in terms of diagnosis and treatment of the pathology. This program can be expanded to further diseases and even more systems to help the user in more situations where the symptoms are related between different diseases and it can be hard to tell by firsthand what the patient have.

Another advantage that I saw during the tests was that the user can easily copy and paste all the text in the command line to a text file and save it for future examination or just as evidence of his or her work.

**Setup Instructions**

First the user has to consult the recipes.pl file in order to use the program. Once the file is consulted the user has to type the following function with this order:

*get\_all\_recipes([Ingrediens],R).*

Where Ingredients is a list of ingredients for example: [water,sugar,orange] and R is the result and can be anything you want as long as it starts with capital letter for example: Result.

If the user would like to expand the knowledge base, it can be done by editing the file “recipes.pl” in the first part, it has a header named ‘Knowledge Base’. Each recipe registry must follow the format:

*recipe(Name of the recipe,[‘ingredients list’, ’comma separated’]).*

On each section of the recipe registry, the user must enclose the text that consist in more than one word between simple quotes.

Once the user has finished editing the knowledge base, he can consult the recipes.pl again and the changes should be shown.

**Evidence or references to back up claims.**

All the evidence of the test execution is contained in the ‘tests.csv’ file. This csv is the result of the 30 people that help me to fill the google form after using the What to Eat? App, the graphic results are the ones I showed in the results part.

The GitHub page of this project is: <https://github.com/Salomon21/What-to-eat->