# Business Scenario

A medical company has demanded the development of software that will be used for dietary support. Our team of four software developers was formed to produce a program which would allow a user to input data on the meals they plan to eat and would output suggestions that correspond to the user’s goals (e.g., weight gain or loss), health issues (e.g., allergies, health conditions) and eating preferences. This will help the user ensure that they are eating well and within their daily calorie intake and food restrictions that they set.

To begin using the program the user will need to create an account which will allow them to log in. An account system will exist so that the data on every person’s health and nutrition will be stored privately on their account alone.

The app will allow a person to create their own recipes by inputting the ingredients and writing notes on the methods of cooking it. When the recipe is created, its nutritional details such as calories and macro-nutrients will be calculated. The app will calculate the amount of ingredients needed to produce a selected number of servings of the recipe. This recipe will be stored within their account. It can then be viewed in their list of recipes.

The other use of the app will be tracking the food intake of the user. The user will be able to add a meal to the current day’s record and the system will sum up the calorie and nutrient intakes for the day. If a recipe that the user adds exceeds the suggested intake, the app will suggest alternative dishes or help substitute or remove non-core ingredients. However, these will only be suggestions and won’t be strict and annoying to the user as there will be cases where the user will want to cook/eat irregularly.

Macrit will reduce the amount of time needed to think about how many and which ingredients to buy and avoid calculating their food intake manually.

# Software Lifecycle

The software lifecycle model that we will follow is very important to decide on before starting the development process as this will decide on how we allocate our time, the order of our tasks and how we will work as a synchronised team. The conditions that were placed on us were as follows:

* We are to produce a design document before developing the app
* We only have 5 weeks (from week 7 to week 12) to work on the implementation.

This created a dilemma about whether to follow a Waterfall model or an Agile process. We would have already finished the requirements and the documentation, so it was a good opportunity to use the Waterfall model. This is a step-by-step structured process where each step is done before the next one and there is no back-tracking.



Waterfall Model [1]

However, this would mean that there would be no room for correction and a lot of time would be spent on each step to ensure we won’t change anything in the future. This was a risk we didn’t want to take because we didn’t have a lot of time to do this. Instead of this we favoured the Agile approach because that would mean we would be able to concurrently work on different parts of the project.

We liked the idea of CICD (Continuous Integration Continuous Development) because that would maximise our productivity and communication. We will have regular stand-up meetings to make sure we are in sync and are aware of each team member’s progress and to integrate the different parts of the software (e.g., UI to backend, and modules will always have updated unit tests). The only catch is that we might revisit code too often and keep adding features, this is called scope creep. We will try to minimize this as much as possible.

Diagram

Description automatically generated

Agile Model [2]

# Quality Attributes

There are certain quality attributes that we prioritize when we are developing Macrit. The main ones are reliability, extensibility, and maintainability.

The application will be reliable because we plan to use a food API that will already have all the data we need. This data was already used and tested by many other applications, so we are confident in its accuracy. This means the user can rely on the information that our app provides and won’t have to go to third party sources to change any data. The use of this software should be effortless and should require no extra research.

The extensibility of the app is very important because there will always be room for change and improvement. The feedback we get from users will create a need for further development of new features to satisfy their needs. Every module should be reusable so that it can be used almost like a library. Refactoring the code shouldn’t mean re-developing completely. The implementation of functions should allow for the addition of possible new ones.

Unit tests will be written to ensure that the app is maintainable. We will be working in an agile way so anytime units of code will be changed we will also change their tests. A Jenkins pipeline will also be setup to make sure a test build is done every time we push to the GitHub repository. This way we will know if the program fails at the build stage and if the tests are passed. Of course, manual tests for the UI will also be necessary because not everything can be unit tested, we need to make sure that everything also works on the frontend level.

# References

[1] *SDLC - Waterfall Model* (2022). Available at: https://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm (Accessed: 1 October 2022).

[2] (2022) *Asana.com*. Available at: https://asana.com/resources/agile-methodology (Accessed: 1 October 2022).