OLAF

Project Dossier

Aurélien Fernandez

Table of content :

1 - Introduction

1.1 - Purpose of this document

This document serves as a report containing the considerations, the decisions and conclusion of my graduation project.

1.2 - Project representatives

This project is impacted by multiple groups with distinctive interest, there groups are:

Myself, Aurélien FERNANDEZ. I am the project owner, currently studying at ALGOSUP for the fifth and final year. This project is the result of my studies, leading to the diploma provided by ALGOSUP.

ALGOSUP, a programming school based in Vierzon, in France.The school provides guidance to the project owner and provides an official review after a formal presentation at the end of the project.

Group Koncept-Komet, a company of 55 employees based in Toulouse, in France. It offers digital solutions along with advisors for companies. This company is offering this project an environment named Amazon Web Services.

1.3 - Milestones

Dates	Milestones
30/08/2024	The main branch of the GitHub repository is frozen for the first review of the project.
13/09/2024	First presentation of the project in front of the school and external reviewers to assess the quality of the project.
To be defined	The main branch of the GitHub repository is frozen for the second review of the project.
To be defined	Second presentation of the project in front of the school and external reviewers to assess the quality of the project.

2 - Project definitions

2.1 - Business story

2.1.1 - Context

In France, most of the population possess a garden or at least a plant in a pot. This study from the IFOP and the UNEP shows that 63% of the French population have a garden, 58% of these gardens are private and 5% are shared between multiple people/families. This study also shows only 42% of Parisians are owning or participating in the maintenance of a garden. Finally, 70% of the participants grow either vegetables or fruits to sustain themselves or to reduce the cost of food.

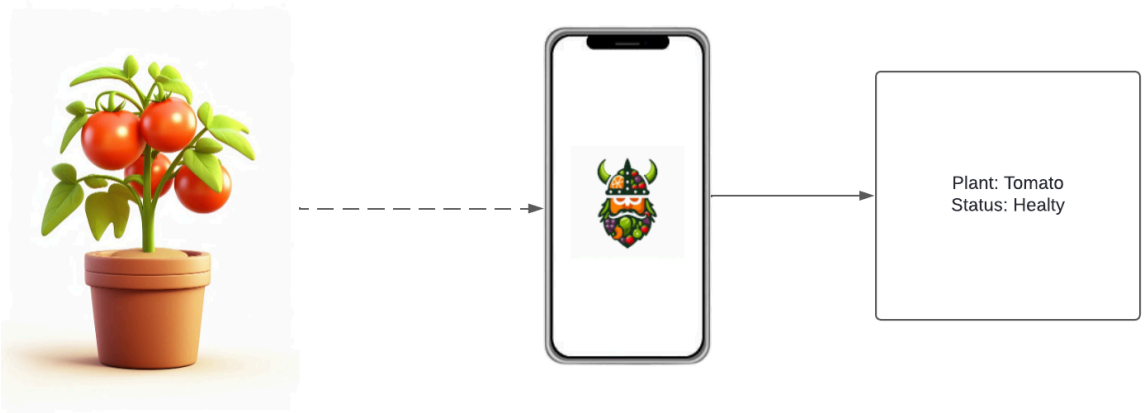
Learning about this, a question arose in my mind, what can I do with this information? Then I learned that people couldn't recognize a plant disease and in consequence deal with a disease accordingly. This is how the project named OLAF began.

2.1.2 - The idea

My idea is OLAF, which stands for OnLine Automated Farm. As the name suggests, it is an automated farm or more precisely an automated plant pot linked to a mobile app to detect when a disease or a virus is infecting a plant.

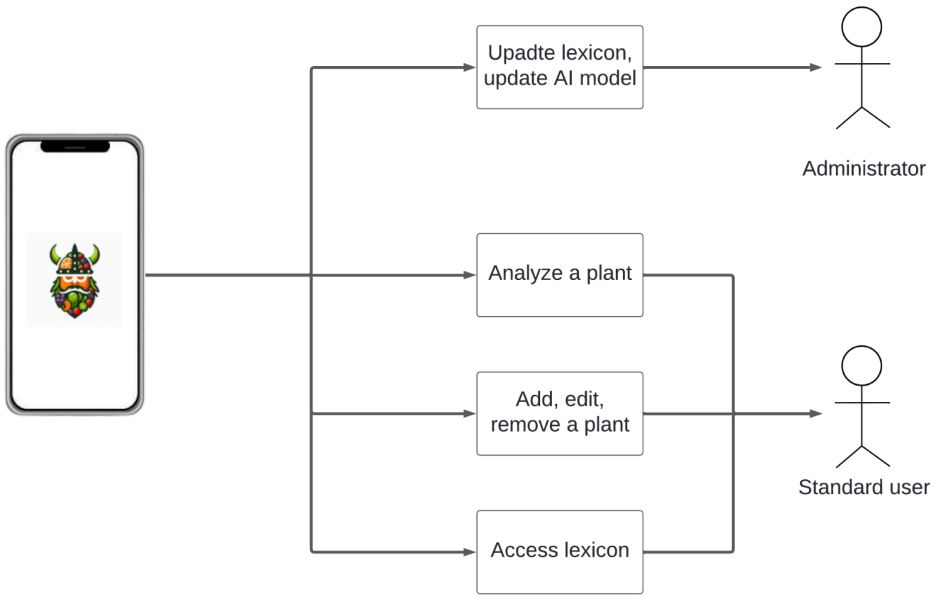
To achieve this, an AI can be used to analyze an image and detect specific patterns linked to a disease or a virus.

Additionally, the project can be divided into two sub-projects:the mobile app with the AI and an automated plant pot connected to the user's account on the mobile app.



2.2 - Functional view

2.2.1 - System in context



2.2.2 - User audience

The targeted audience could be defined to anyone maintaining a plant, more particularly vegetable.

In France, using the numbers from the study mentioned earlier, it represents at least 42,455,700 people (63% of a population 67.39 million as of 2019).

The targeted audience could be increased by internationalizing the application.

Please notice that the targeted audience includes only people who are of legal age

2.2.2.1 - Personas

Alexandra Davies

The patient one

79 years old

My garden is my little paradise

Goals	Frustations	Personality
- Fill her garden with flowers and small fruits - Show off her gardening skill to her friends	- She is not really interested in the new technologies and needs a simple interface to use an app - She is never sure if she watered her plants enough	- Cheerful - Compassionate

Mia Jackson

The Student

24 years old

I want people to understand a plant's need

Goals	Frustations	Personality
- Raise awareness about plant diseases - Study the stages of a plant's diseases	- Her family keeps throwing plants to the trash instead of curing them - Some diseases are nearly invisible for the human eye	- Studious - Willful

John Tatle

The passionate

21 years old

I want to know why my plants are dying.

Goals	Frustations	Personality
- Grow his own vegetables for himself and his family - Keep his plants healthy	- John knows nothing about plant diseases - Every time a plant is suffering from a disease, he doesn't know how to cure it	- Outgoing - Generous

2.2.3 - Use cases

2.2.3.1 - Mobile application

A mobile application is a software that can be installed on mobile devices. The purpose of this one is to be able to follow the state of your plants or request the analysis of a leaf to detect potential diseases or viruses.

The main use cases for the mobile application are :

- Being accessible for any user once the authentication form is filled,
- Pulling a lexicon from the server,
- Displaying a lexicon describing either a plant or a disease,
- Taking pictures and detect diseases and viruses on a plant,
- Display the result of the analysis of the given picture,
- Possessing at least 2 languages: English and French,
- Changing the language of the app,
- Communicating to the server to access the different data, lexicon or user data,
- Differentiating the type of plant and its current status(healthy or infected) with a precision of at least 95%,
- Adding a plant to an account,
- Removing a plant from an account,
- Editing a plant from an account,
- Displaying an history of the picture taken with the camera along with their result,
- Removing a picture from history.

2.2.3.2 - Web application

A web application is a software available via browsers such as Google chrome, Bing, Opera, Mozila and many more. The purpose of this web application is to add, remove or edit elements in OLAF’s lexicon and to update the AI model.

The main use cases for the web application are:

- Being available only for users identified as administrators,
- Modifying the lexicon,
- Modifying the AI model
- Updating the role of a user, elevating a user to administrator or removing administrator access,
- Pulling an update from the server
- Uploading the changes directly into the server

2.2.3.3 - Embedded system

An embedded system is a specialized computing system designed to perform specific tasks. The purpose of the embedded system of this project is to monitor and manage your plants and detect potential diseases or viruses automatically.

The main use cases of the embedded system are:

- Collecting data (temperature, soil humidity) in real time,
- Reacting to the collected data to benefit the plant (watering),
- Taking images of the plant's leaves,
- Sending the images to the server to detect infections,
- Alerting the user in case of infection.

2.2.3.4 - Out of scope

Some use case referenced will be out of the scope :

- The detection of an infection in plants that cannot fit in a plant pot
- The collection of fruits/vegetable, as it would require a robotic arm

2.2.3.4 - Nice to have

Some use case referenced are nice to have, if there is any possibilities they will be implemented :

- The mobile application can be used on multiple platforms: Android, IOS
- Make the system energetically autonomous : use a solar panel and a battery
- A moving camera, as it would increase the probability to detect an infection if a plant is infected or to follow multiple plants.
- The app is translated in other languages than english and french

2.3 - Design constraints

Types	Constraints
Energy and ecological constraint	The energy consumption of the pot must not exceed 10W.
User experience constraint	The app must be responsive to deliver a clear and understandable interface.
Economical constraint	With limited funding, the project costs are covered directly by the project owner, which constrains the choice of technologies and solutions available for completing the project.

2.4 - Architecture

2.4.1 - Drivers

Drivers
Limit the spending related to the project as much as possible
Provide an app that can be used by anyone, regardless of location or time
Create an architecture which can be easily maintained and easily modified
Ensure the security of user's data

2.4.2 - Non-functional requirements

The project must also achieve multiple non-functional requirements:

Requirements	Definitions
Accessibility	The planter and the mobile app must be designed to be accessible to all users, including those with disabilities. This includes ensuring the app accommodates color-blind individuals and those who are unfamiliar with technology.
Updates frequencies	The mobile app must update the planters data with a frequency of 2 per hours minimum
Mobile app & desktop responsiveness	The app must be responsive to scale with the user's phone/computer size
Mobile app & desktop reaction time	The time between a user's action and the app/desktop app response must not exceed 0.01ms
Maintainability	The overall project must be easily maintainable, to ensure an easy way to update the project
Security	The project must be secure, no one other than the administrators must access user's data or have access to the servers

2.4.3 Resource allocation

The project being without any funds, the funding of the different pieces of the project will be provided directly by the project owner

Additionally, the company Koncept-Komet allowed me to use an AWS environment with two conditions:

- Limit the cost of the environment to 30 € per month,
- Present the project in front of representatives of the company.

2.4.4 Law and regulations

The laws this project has to follow are relatively simple, most of the laws impacting this project are under the GDPR, which includes:

- The user must consent before collecting any data,
- Ensure a secure way to store user data,
- Ensure the user knows which data the project is using,
- Ensure the user knows how the project uses the provided data,
- Ensure the user can access and delete all data linked to the user's at any time.

3 - Application design

3.1 - Colors

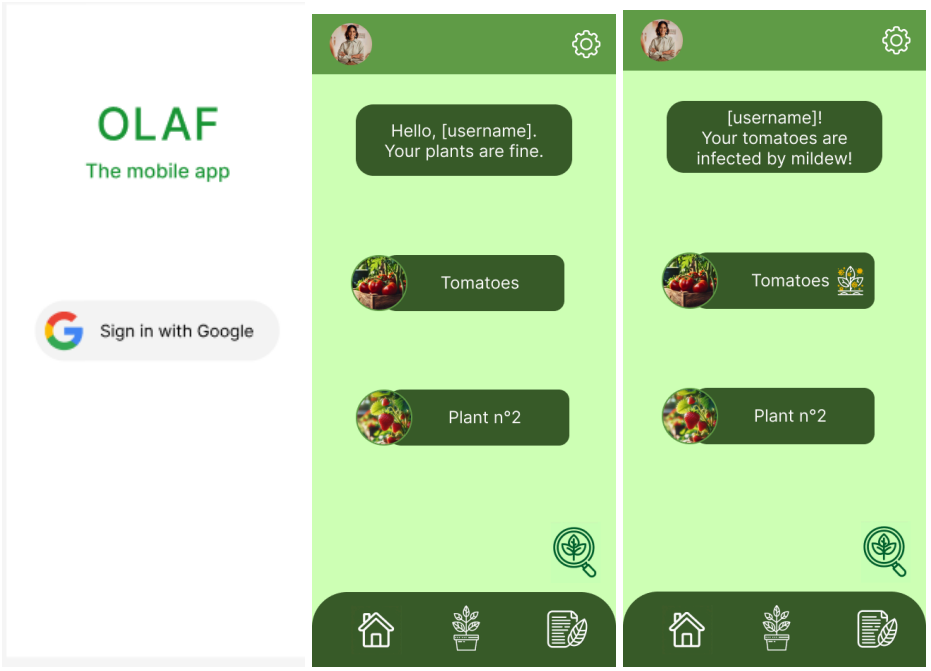
The app will possess a theme with the colors being mainly based on green-like colors: (in hexadecimal):

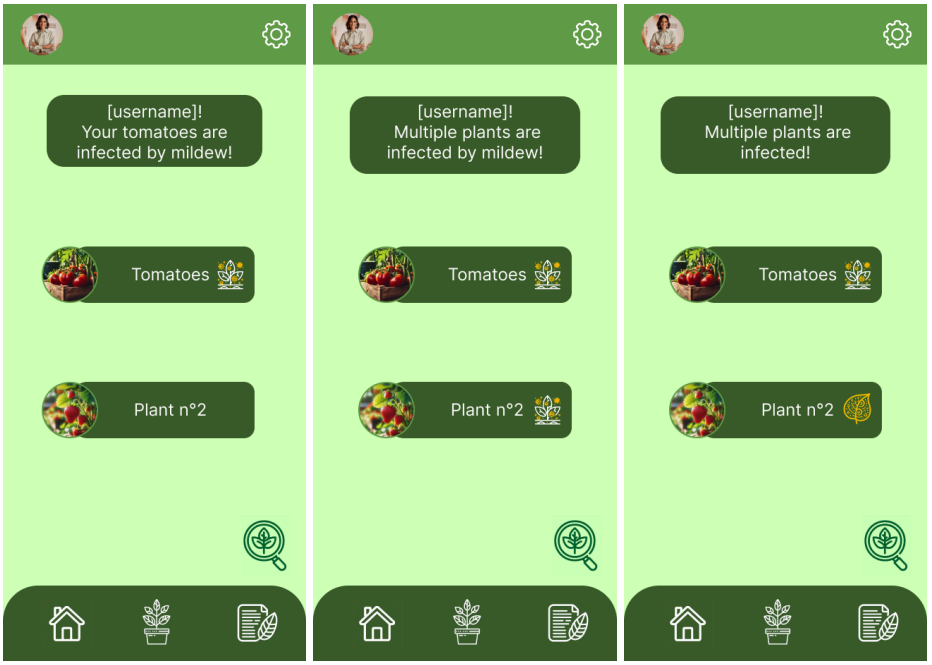
Main color	#395B29	
Secondary color	#629B47	
Background color	#CEFFB7	
Text color	#FFFFFF	

The icons are all outlined icons. The icons used for navigation are only in white and the icons representing diseases are in white and yellow.

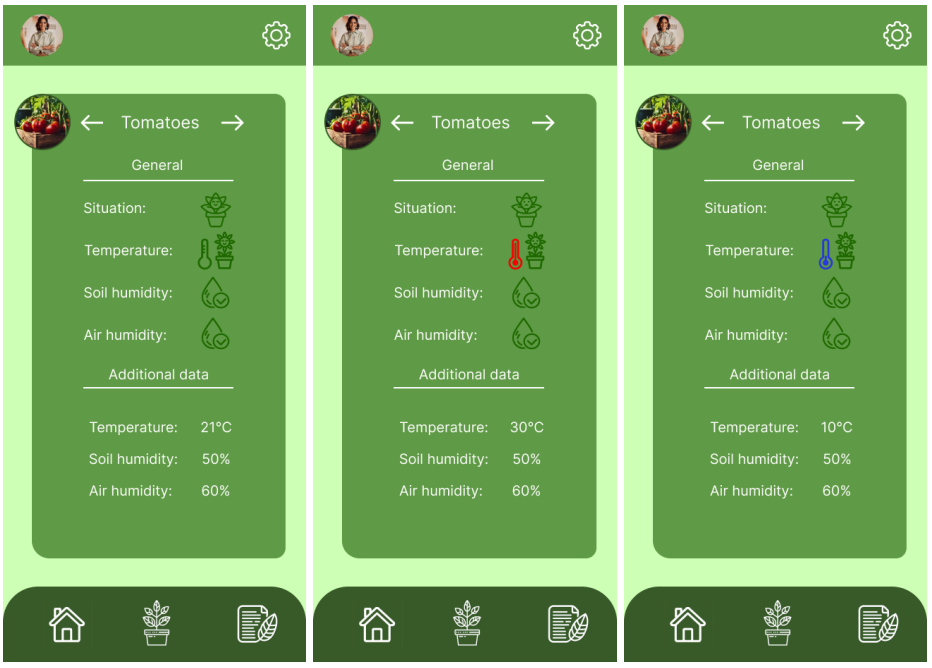
3.2 - Original design

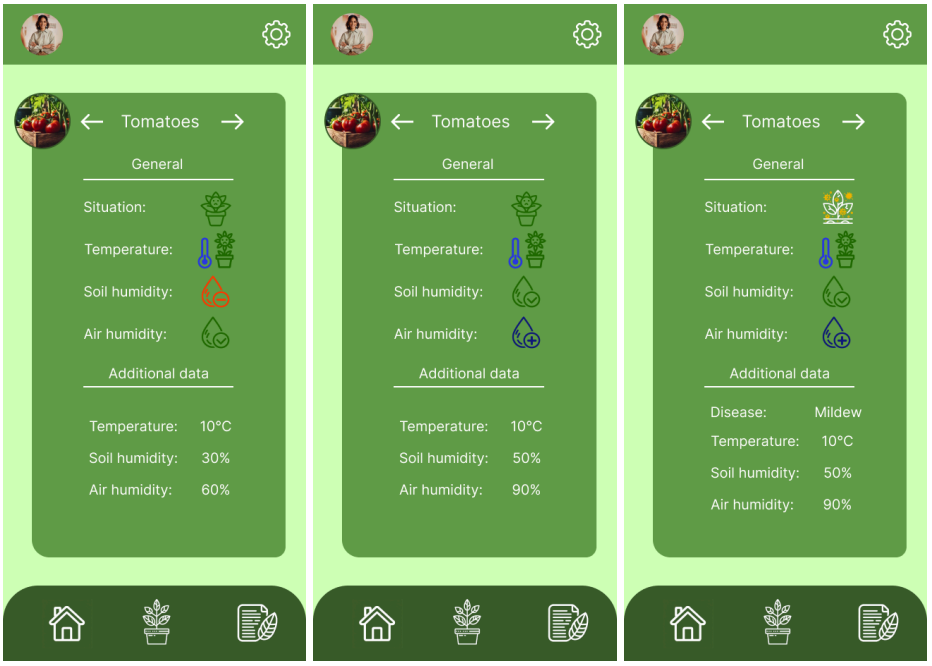
The first page is the connection page, it will appear only the first time or after the user has disconnected his account. If the app has been opened for the first time, it shows a small window to set up the color theme, to permit color-blind people to see the app correctly.



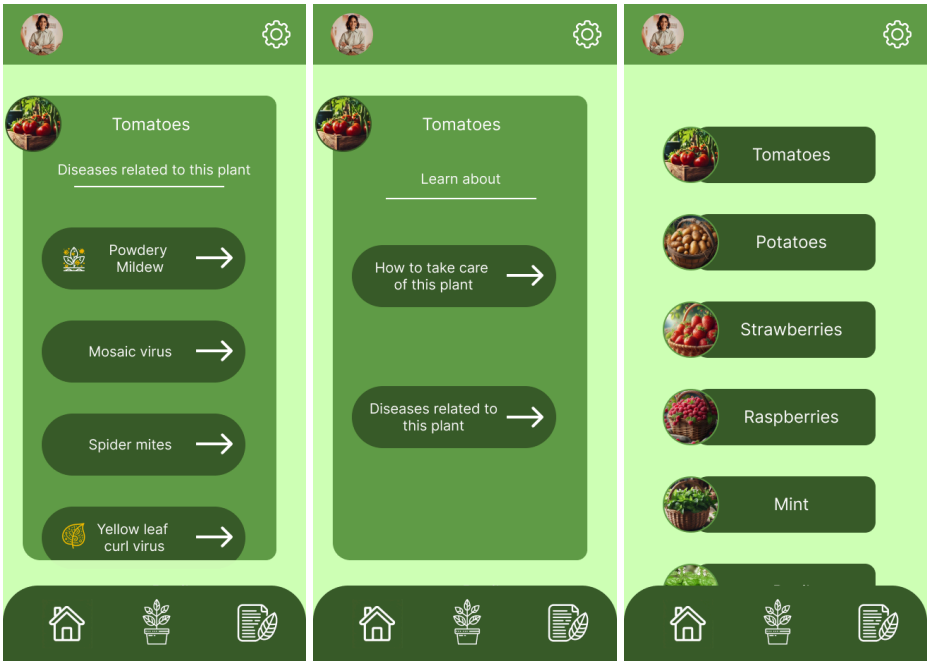


Then, there is the "plant's status" page, which displays the status and information of a plant linked to the account. Users can see the data collected by the planter's sensors, including the ambient temperature, the soil and air humidity and the plant's current state, if it is infected or not.

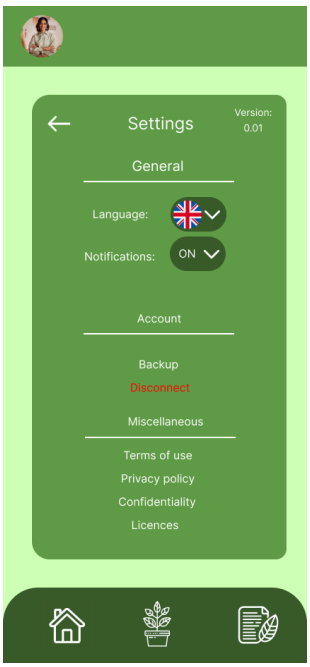




The third page is the lexicon, which allows users to learn about plants and the diseases related to each plant. It shows users images of the disease and how to prevent/cure a disease.



Finally, there is the settings page, with multiple settings such as account management, language and other pages about confidentiality, terms of use and licenses.

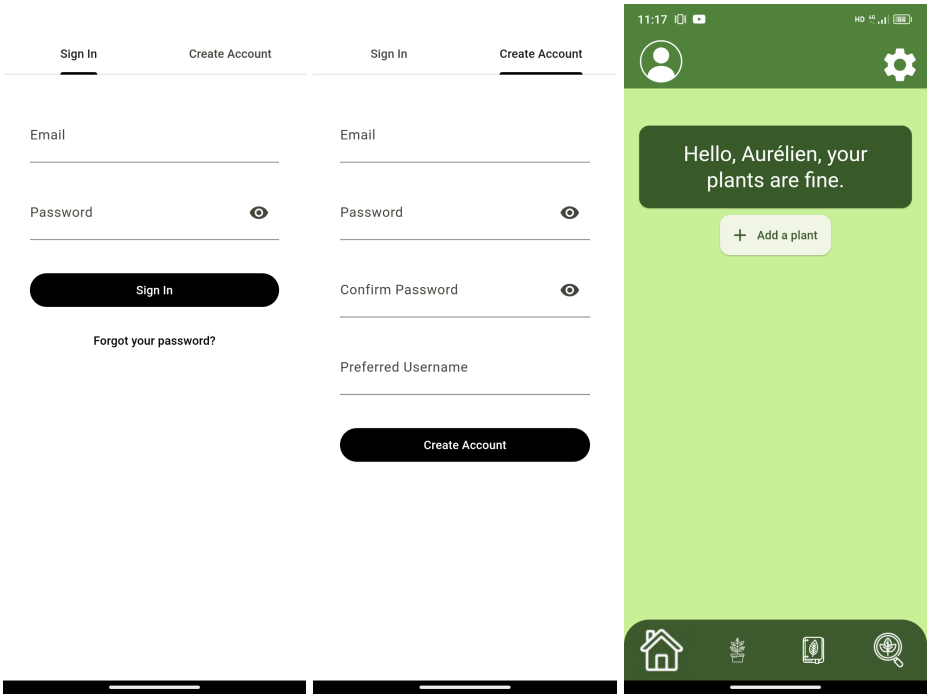


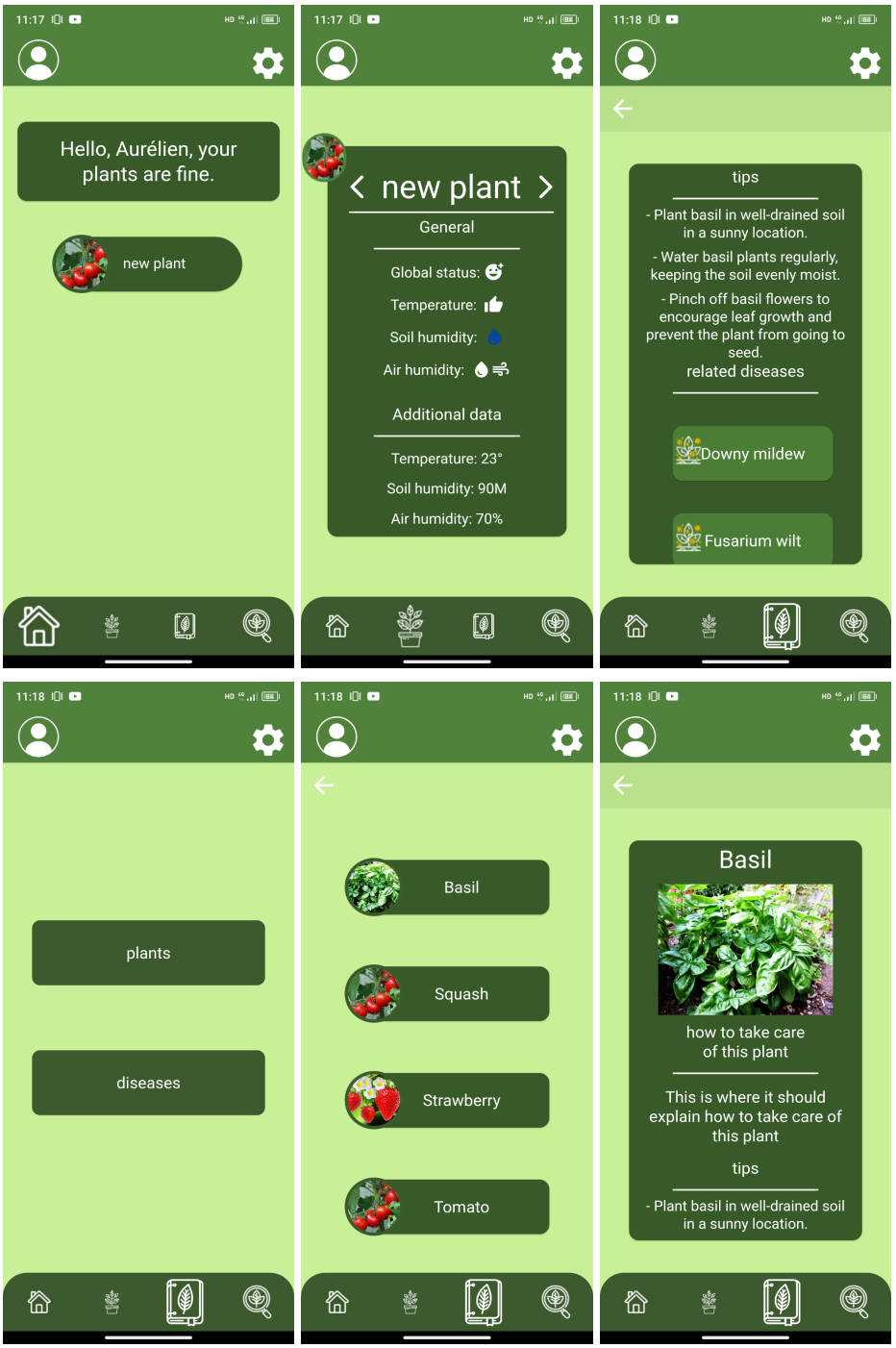
3.3 - Current design

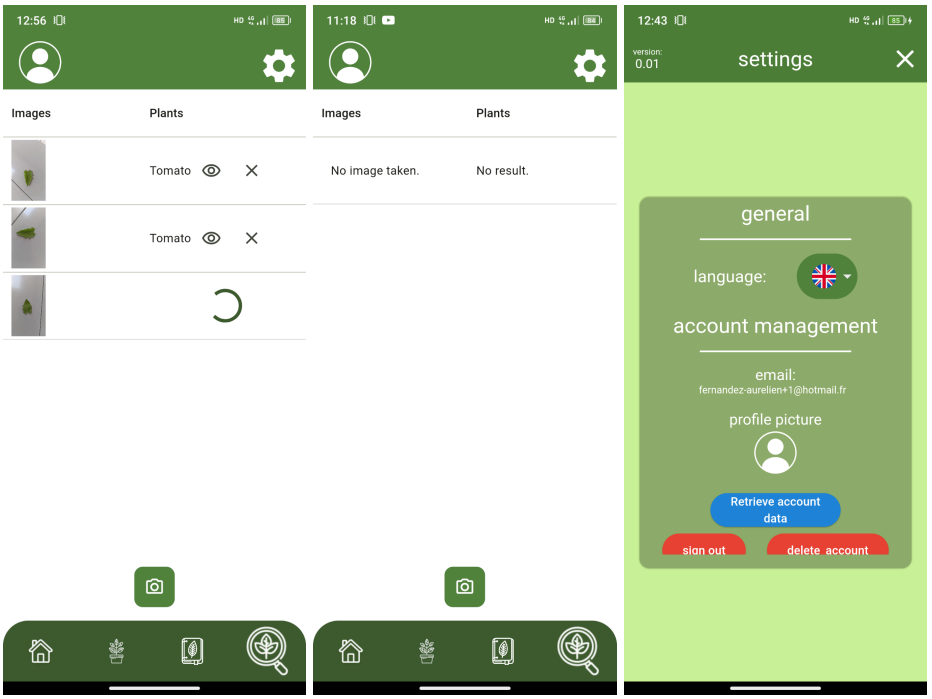
The current design is not entirely different from the one created at the beginning of the project. It has been updated due to the constraints linked to the different devices, user feedbacks and the technologies used.

The current design can also be found at this link: mockup.

The following order is identical to the user path presented in 3.2 - Original design.




11:17
Hello, Aurélien, your plants are fine.
new plant
11:17
new plant
General
Global status:
Temperature:
Soil humidity:
Air humidity:
Additional data
Temperature: 23°
Soil humidity: 90M
Air humidity: 70%
11:18
tips
- Plant basil in well-drained soil in a sunny location.
- Water basil plants regularly, keeping the soil evenly moist.
- Pinch off basil flowers to encourage leaf growth and prevent the plant from going to seed.
related diseases
Downy mildew
Fusarium wilt
11:18
plants
diseases
11:18
Basil
Squash
Strawberry
Tomato
11:18
Basil
how to take care of this plant
This is where it should explain how to take care of this plant
tips
- Plant basil in well-drained soil in a sunny location.



4 - Technical description

4.1 - Development rules

4.1.1 - Repository management

4.1.1.1 - Commit conventions

In a GitHub repository, maintaining an easy-to-navigate structure is crucial. To facilitate locating a specific commit, it is important to fully utilize the key elements of a commit. Each commit in GitHub can be divided into four essential elements:

- **The hash:** A unique identifier generated automatically for the commit, represented by a long string of characters. Typically, the first 7 characters (e.g.5bd350g) are used to identify the commit.
- **The title:** A brief, one-line summary of the commit's purpose. It serves as the name of the commit.
- **The message:** A more detailed description that explains what the commit includes and why the changes were made.
- **The content:** The actual changes made in the commit, including files that were added, modified, or deleted in the repository.

The conventions used in this project are heavily inspired by this document: Conventional commits.

4.1.1.1.1 Title

The title is one of the most important elements in a commit. It serves to describe what the commit is about and provides a quick summary of what is implemented, fixed or removed. As such, it has to follow a few rules.

It has to begin with one of the following:

- "FEAT:" if the commit implements a feature to the project.
- "FIX:" if the commit fixes a bug encountered after an implementation.
- "CHORE:" If the commit is about a routine task, for example, refactoring, it is mostly used for tasks that are neither features nor fixes.
- "DOCS:" if the commit is about changes to a document or adding documents.
- "PERF:" if the commit is about optimizing the performances.
- "REMOVE:" if the commit removes a feature.

Furthermore, a title shouldn't be more than 50 characters long and must describe the action of the commit, a verb, and the name of the implementation or the name of the file involved.

4.1.1.1.2 Body

The body describes the commit with more details, and as titles, it has to follow specific rules:

- Provide context: explain why this change is needed.
- If possible, summarize how changes were applied, notably for implementations, fixes, refactoring, or optimization.
- Reference issues if needed.
- Avoid long paragraphs. Make a list of what has been done.
- No jargon without explanations. Other contributors must understand commits.

The body can also contain specific keywords that can interact with the repository, such as adding co-authors, closing issues, etc. They can be found here in the footer section.

4.1.1.1.3 Examples

For a feature:

```
Feat: implementation of multiplications

Implement a new feature to support multiplication operations in the
calculator.

- Implementation of the multiplication algorithm.
- Update the user interface to include multiplication.
- Add unit tests related to multiplications.

This feature enhances the usability of the calculator, providing users
with a new set of operations.
```

For a fix committed after the previous example:

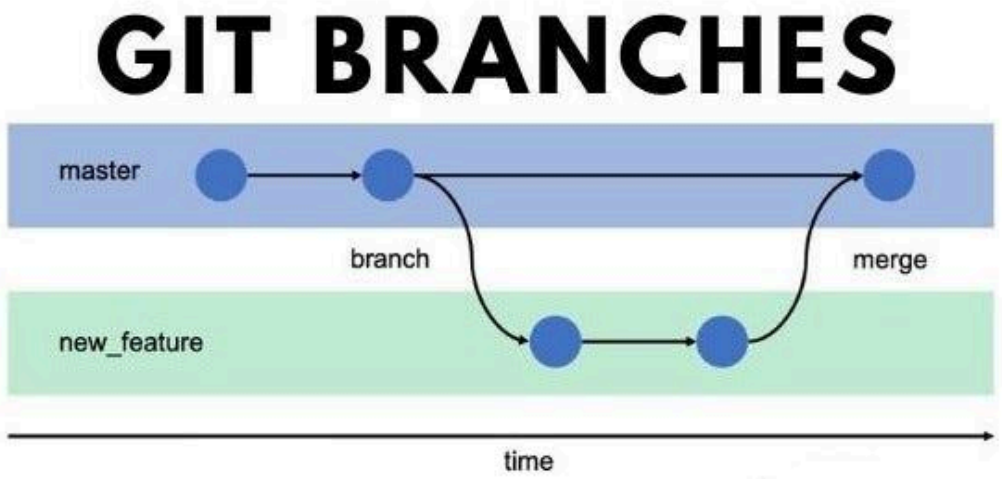
```
Fix:  overflow with multiplications

Fix an issue where the multiplication may create an overflow if the
inputs are higher than expected (over a trillion).

see commit: 1a2b3c

- If one of the values is higher than the limit, divide it into two
separate values.
- If a value is too high, represent it with exponent (1e10 is equal to
1x(10x10))

This fix allows users to apply multiplications to higher numbers
without being subject to an overflow.

Closes: #324
```

4.1.1.2 - Branches

Branches are one the of the key elements of Git, it allows members of a project to add, modify or remove files or parts of files without interfering with the work of another member. For example, a document present in the branch of the same name, wouldn't appear in the main branch, unless a merge has been made.

- **main**: Where the official version of the project is stored along with the documents.
- **dev**: Where the code of the project is stored. It is moved into the main branch when the required features are finished and no known bug is present in the code.
- **documents**: Where all documents are stored, awaiting to be pushed into the main.

Hereunder you can find a representation of Git's branches system:



4.1.2 - Code management

4.1.2.1 Naming conventions

Dart, the programming language used for this project is using strict rules regarding the naming of the different elements composing a project, here is the list of these rules:

- **Files and Folders**: Names of files and folders should use **snake_case**. This means that if a name contains more than one word, the words are separated by underscores. For example, `my_project_folder`.
- **Functions**: Function names should use **camelCase**. In this style, multiple words are written together without spaces, and each word after the first one starts with an uppercase letter. For example, `calculateTotalSum`.
- **Classes**: Class names should use **PascalCase**, where each word, including the first one, starts with an uppercase letter. For example, `UserAccountManager`.

4.1.2.2 - Comments

Another way to ensure the ease of maintaining a project is to add comments. Comments help improve code readability and make it easier to understand and maintain.

4.1.2.2.1 - Import comment

As Dart implies the use of multiple libraries, the use of import comment can be important to differentiate which import is created specifically for this project or comes from libraries. Here is an example from the file main.dart:

```
//------------------- CUSTOM IMPORTS --------------------
import 'package:olaf/amplify_configuration.dart';
import 'package:olaf/app_localization.dart';
import 'package:olaf/cache/shared_preferences%20.dart';
import 'package:olaf/home/home_page.dart';
import 'package:olaf/cache/loader.dart';
import 'package:olaf/settings/save_settings.dart';

//------------------- FLUTTER IMPORTS -------------------
import 'dart:async';
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:flutter/services.dart';

//------------------- AMPLIFY IMPORTS -------------------
import 'package:amplify_auth_cognito/amplify_auth_cognito.dart';
import 'package:amplify_authenticator/amplify_authenticator.dart';
import 'package:amplify_flutter/amplify_flutter.dart';
import 'package:amplify_storage_s3/amplify_storage_s3.dart';
```

4.1.2.2.2 - Functions comments

Comments can also help in order to be able to update and re-use functions. Here is an example of one of them:

```
///This function creates a copy of a given image to the directory used
to contain the profile picture,
///allowing the app to update the new picture as the download needs a
reload to be used.
///
///[image] is a file containing an image and [user] is the connected
AWS AuthUser.
Future<String> ChangePictureLocal(File image, AuthUser user) async {
```

4.2 - Rational

The choice of technologies was made depending on the factors mentioned in 2.3 - Design constraints and in 2.4.1 - Drivers.

4.2.1 - Devices

- For standard users, the decision to create a mobile app for phones was driven by the second driver, as most people own a phone, and it can be used nearly anywhere.
- For administrators, the decision to create a desktop app was driven by the need to send files, specifically a file containing the AI model. Additionally, separating the administrator app from the user app enhances security by ensuring that the functionalities are not combined in the same application, reducing the risk of code-related security leaks.

4.2.2 - Programming language

The choice of the programming language was driven by the need to create both a mobile app and a desktop app. This led to the programming language Dart along with its SDK Flutter, which can create an application for Windows, MacOS, Android and IOS from the same source code.

4.2.3 - Infrastructure

The choice of backend was made due to multiple needs:

- A service available everywhere at any time,
- A service that can handle a large number of user at the same time,
- A secure storage, to avoid security leaks,
- A service capable of creating and handling a user pool.

These needs lead to the decision to use Amazon Web Services. As it offers services handling these needs. AWS also has the advantage of being cheap, which was required due to budget constraints.

4.3 - Artificial Intelligence

4.3.1 - What is an artificial intelligence?

An artificial intelligence, is a system or machine designed to perform tasks that usually require human intelligence. These tasks can include understanding natural language, recognizing patterns, making decisions, solving problems, and learning from data. Artificial intelligence can be used in many fields such as : image classification, virtual assistant, autonomous vehicle and more.

4.3.2 - Why use artificial intelligence?

In this project, the use of artificial intelligence is crucial, as it is the only way other than the human eye to detect diseases from an image.

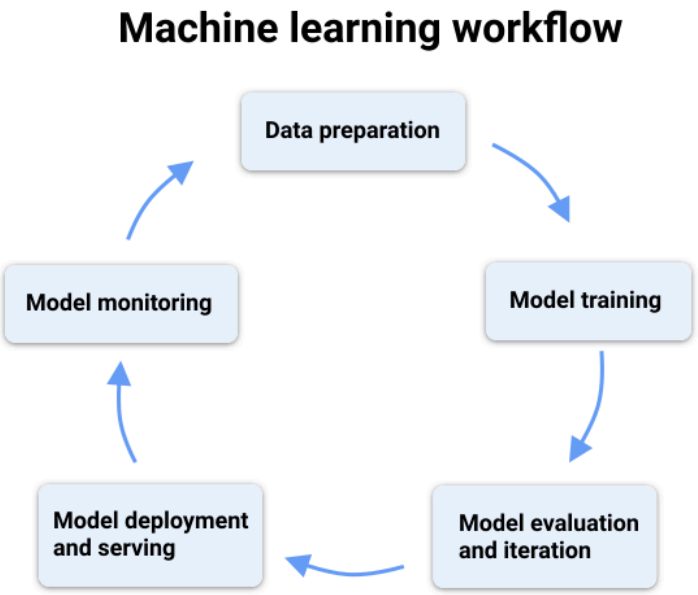
For this project, the artificial intelligence used is designed for image classification, which can be defined as an artificial intelligence trained to analyze an image and specify in which category the image belongs. For example an AI trained to differentiate cars from bicycles could classify cars in the first category.

4.3.3 - training an artificial intelligence

Training an artificial intelligence requires multiple elements:

- A consistent amount of training data
- A equivalent amount of validating data
- A model composed of neural networks

The process of creating an artificial intelligence can be represented by a simple loop as this:



As represented on the precedent page, to create an AI, the most common way is to prepare data to train the model, then to train the data, at the same step the model will be evaluated automatically, giving a success rate in percentage. The next steps are deploying artificial intelligence to use it and monitor it.

The data preparation can be divided into three steps:

- The collecting of data,
- The filtering of the data,
- The modification of data, for example for images, modifying each image by changing colors, rotation and scale increases the quantity of the training data available.

Then the training is composed of a four steps, which are autonomous:

- The model will start a cycle, also named an epoch,
- During this cycle, the model will try to classify images,
- The result of the model will be compared to the same data but classified correctly,
- The success rate will be returned in percentage and the model will start a new cycle.

4.3.4 - The difficulties of training an artificial intelligence

The difficulties of training an artificial intelligence are numerous, here is a list of the most common risks:

- The lack of training data,
- The excess of training data,
- The presence of undesired training data,
- A lack of balance in the training data,
- The lack of epochs,
- Overtraining the model,
- The lack of training cycles.

4.3.5 - CropAI

4.3.5.1 - PlantVillage

CropAi is the name given to the AI used in this project. CropAI is trained over the data provided by the PlantVillage dataset. The dataset has been reduced for the purpose of this project as some data weren't necessary.

Here is a list of the plants and the corresponding statusCropAI is currently trained on:

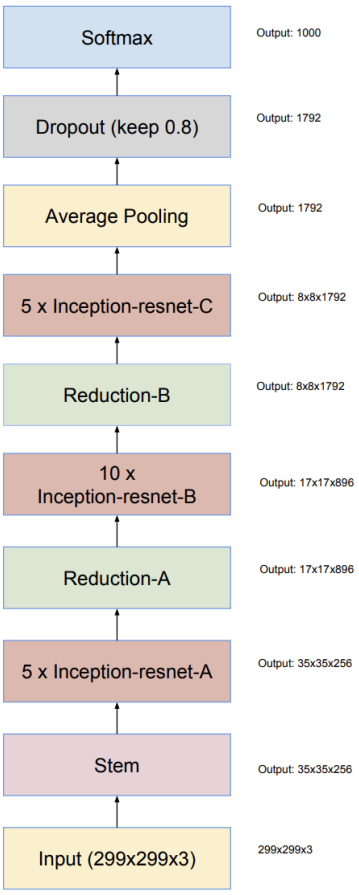
- Tomato
    - Healthy
    - Bacterial spot
    - Early blight
    - Late blight
    - Leaf mold
    - Septoria leaf spot
    - Spider mites
    - Target spot
    - Mosaic virus
    - Yellow leaf curl virus
- Pepper bell
    - Healthy
    - Bacterial spot
- Potato
    - Healthy
    - Early blight
    - Late blight
- Strawberry
    - Healthy
    - Leaf scorch
- Raspberry
    - Healthy

4.3.5.2 - ResNetV2

The model used for the creation of CropAI as base is ResNetV2, as it has been heavily trained on image classification it is a good match for this project.

ResNet is a convolutional neural architecture built on the Inception architecture.

Hereunder is the official flowchart of the ResNetV1 and V2:

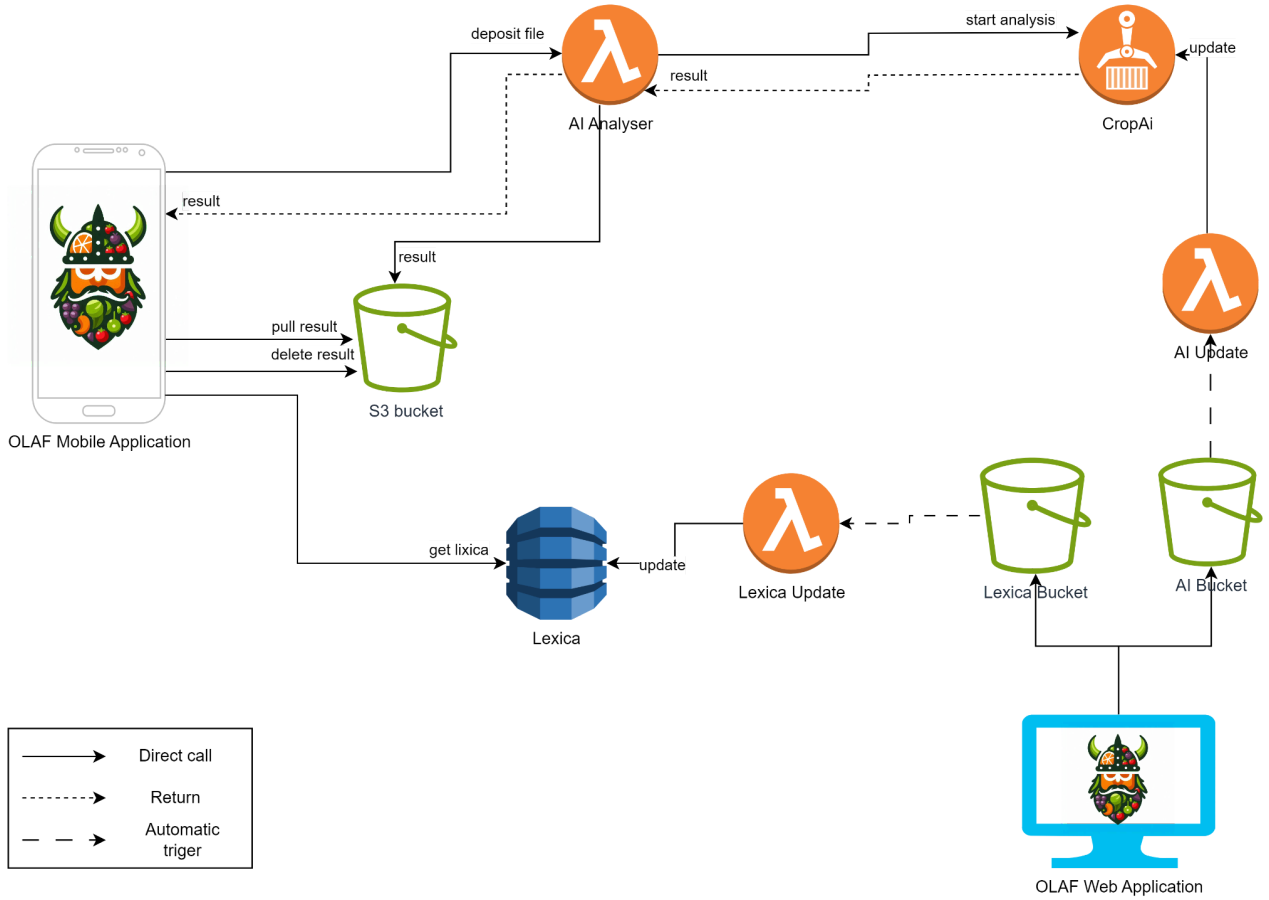


ResNet is known for its efficiency regarding image classification, which made it one of the best choices as the base for an image classification AI.

4.4 - Architecture details

4.4.1 - Overview

Here is the full architecture made for Olaf project :



The components shown in this diagram are explained in more detail in the following chapters.

4.4.2 - S3 storage

Amazon S3 buckets, which means Simple Storage Service, is a scalable cloud storage service designed for storing any data from anywhere. More information about this technology can be found here: What is Amazon S3?

This project only uses one S3 named olaf-s3. This S3 is composed as this:

```
olaf-s3
├── 📁user
│   └── 📁[userID
│       ├──📁analyze
│       │   └──📄[date].json
│       └──📁profile-picture
│           └──🖼️[userID].png
├──📁updat
└──📁mode
   └──📄CropAI.keras
```

This S3 contains 3 mains folders:

- **users**: A folder containing a sub-folder dedicated to a user, they are named after the id of their dedicated user.
- **update**: A folder used to update the lexicon and the AI model.
- **model**: A folder containing ai AI model.

Each user folder contains two folder:

- **analyzed**: A folder containing a json named after the creation timestamp, it contains 2 elements: the image taken by the user, stored in base64 along with the result of the AI.
- **profile-picture**: A folder containing the profile picture of the user.

Finally, the update folder is made to temporarily store either a json file named "update" or a Keras file named CropAI.

The json file shall be structured as this:

```
{
    "plants":{
        "tomato": {
            "name": "Tomato",
            "howTo": "A description on how to take care of the plant",
            "image": "either a link or a base64 image",
            "tips"[
                "A tip on how to take care of the plant",
                "A tip on how to take care of the plant"
                ]
            "diseases":[
                {
                    "name:" "Powdery mildew",
                    "image": "A link or a base64 image representing the
plant with said disease"
                }
            ]

            }
    },
    "diseases": {
        "Yellow leaf curve": {
            "name": "Yellow Leaf Curve",
            "image": "either a link or a base64 image",
            "icon": "either a link or a base64 image",
            "description": "a description of the disease",
            "prevent": "A description on how to prevent the disease",
            "cure": "A description on how to cure the disease if
possible"
            }
        }
}
```

Note that in this json file, if no disease or plants needs to be added or updated from the lexicon, the category is not necessary. For example if an administrator wants to add a single plant, the "diseases" part can be completely removed, leaving only the plants part.

4.4.3 - Dynamo database

Amazon DynamoDB is a fully managed NoSQL database service provided by AWS that offers fast and predictable performance with high scalability.This service is used in this project to store the lexicon, containing definitions of plants and diseases.

In this database, each plant and disease has a key value assigned, this key value is the version. The version was created to be able to revert the changes made to an element if need be.
For example:

- The plant tomato has 4 version inside the database,
- An administrator wants to add a tip to the tomato but instead delete all of the tips,
- The administrator reverted to version 4.

This ensures a secure way to modify the lexicon without losing time correcting mistakes made by updating the lexicon.

On the user's part, each time the mobile app is started, the lexicon is fetched and stored temporarily in the cache. This ensures the app doesn't need to send a request each time the user navigates through the lexicon.

4.4.4 - Lambdas

AWS Lambda is a serverless computing service provided by Amazon Web Services called upon specific events without provisioning or managing servers. Additionally, it automatically scales to handle the number of requests. More information about this technology can be found here: What is AWS Lambda?

Here is the list of the lambda used in this project:

name	trigger	effect
lexica-update	Upon the creation of the update.json file in the update folder	Add new versions of the plants and diseases mentioned in the file. Once finished, delete the file in the update folder.
ai-update	Upon the creation of the file CropAI.keras in the update folder	Replace the file present in the folder model by the new one. Once finished, delete the file in the update folder.
image-analyzer	Called directly by the mobile app	Process the given image with the CropAI.keras model. Then create a new json file in the user's "analyzed" folder in the S3. It also sends the result to the mobile app to inform the user of the result.

The flowcharts of the lambdas can be found in annex 1,2 and 3.

4.4.5 - Elastic container registry

An elastic container registry is an AWS service managing the storage and the execution of a Docker image, its advantages are its reliability and high scalability.

This project requires this technology to store the image-analyzer lambda. Due to its use of tensorflow to load the model, the size of the folder containing both the library and the lambda's code weights at least 2 GB which is higher than what a simple lambda can hold. This constraint leads to the use of a Docker image.

5 - User feedbacks

To test the application in real-life situations I had only one choice, find users.I was able to find at least 4 users and each had important important feedbacks to provide:

5.1 - Positive feedbacks

- The interface is clear and easy to understand,
- It is easy to navigate in the lexicon,

5.2 - Negative feedbacks

- The AI is unable to differentiates objects that aren't plant leaves,
- The camera page's background feels not in place as it is white while other backgrounds are green

5.3 - Suggestions:

Suggestions	Has been applied?
Change the camera to place it in the navigation bar and as an entire page with a history showing the multiple pictures and their result.	Yes
Change the displaying of text, by removing the sliding and replacing it by the scaling of the text and the use of multiple lines.	Planned

6 - Conclusion

6.1 - Retrospective

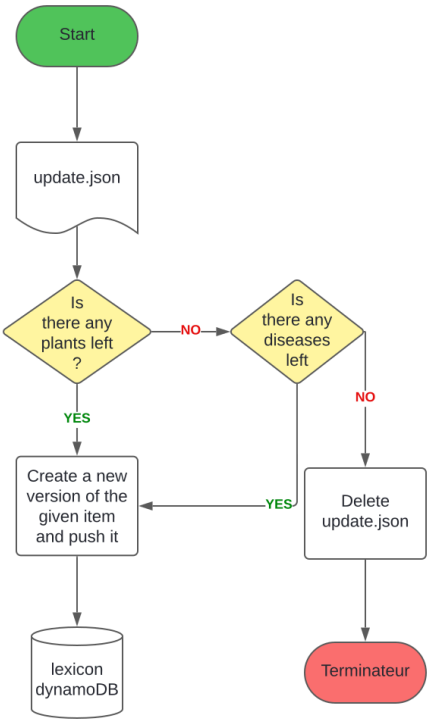
The creation of an application and of an artificial intelligence brought much more insight to me than I first thought. I learned a lot about AWS along with AI and finally I had to deal with multiple roadblocks I first thought impossible to overcome, first of all, the training of my AI took time, multiple hours were needed only to find out the result was unsatisfactory. But in the end, I succeeded in most of the criteria I wanted to succeed. I indeed lacked insight regarding artificial intelligence at the beginning of the project, leading to mistakes, since then, I learned, I understood where I made mistakes, why I couldn't increase the success rate of my AI. I am now able to start this project a second time or even enhance the current outcome without the unclear areas I couldn't apprehend when I started.

6.2 - Acknowledgments

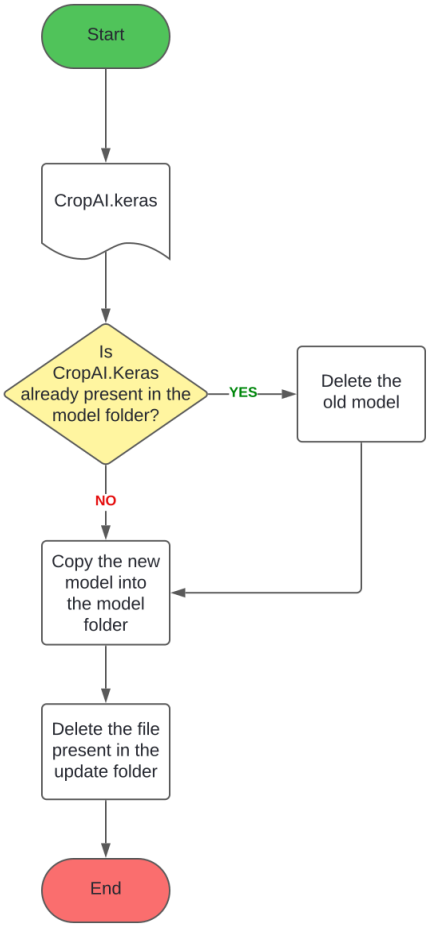
I wanted to thank ALGOSUP for their support and their teaching. Throughout this project and of course my studies I could learn many things related to project management, programming and of course self-management. I also wish to thank Koncept-Komet for their support on this project, the environment provided has been extremely efficient and well-suited for this project. Finally I particularly wanted to thank Sébastien FERNANDEZ, my brother, who helped me to learn about the multiple AWS services and for being the bridge between me and Koncept-Komet. With the help of these participants I could do far more than I could have alone.

7 - Annex

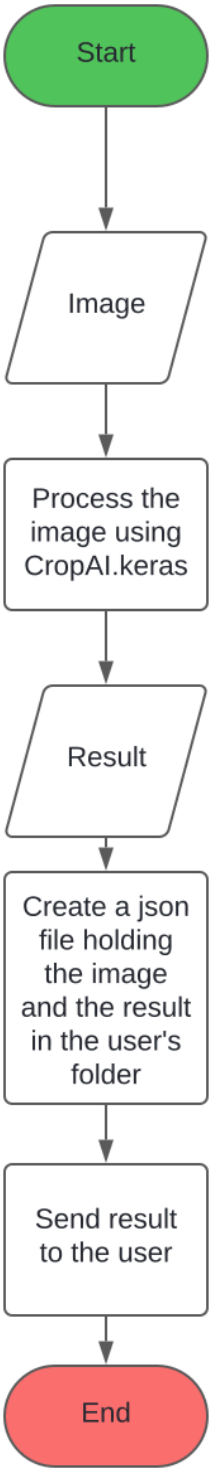
7.1 - lexica-update



7.2 - ai-update


Start
CropAI.keras
Is CropAI.Keras already present in the model folder?
YES
Delete the old model
NO
Copy the new model into the model folder
Delete the file present in the update folder
End

7.3 - image-analyzer



8 - Glossary

Names	Abbreviations	Descriptions
Amazon Web environment	AWS	Amazon Web Services is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis.
Application Programming Interface	API	An API is a way for two or more computer programs or components to communicate with each other. It is a type of software interface, offering a service to other pieces of software.
Artificial Intelligence	AI	Artificial intelligence refers to the field of computer science focused on creating systems capable of performing tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, understanding natural language, and perception.
Dart		An open-source programming language developed by Google, designed for building web, server, and mobile applications.
Docker		Docker is a set of applications used in the virtualization of operating systems.
Docker image		An executable containing the instructions necessary to create a container.
Flutter		An open-source UI framework developed by Google for building natively compiled applications across mobile, web, and desktop from a single codebase.
Git		A distributed version control system used to track changes in files and coordinate work among multiple people. It allows users to manage project versions, revert to previous states, and collaborate

Names	Abbreviations	Descriptions
		efficiently by merging changes from different contributors.
Institut Français des Opinions Publiques	IFOP	The IFOP is an international polling and market research firm based in France.

Names	Abbreviations	Descriptions
General Data Protection and Regulation	GDPR	GDPR is an European Union regulation on information privacy. It is known as the toughest privacy law in the world as it imposes more unbendable rules to companies regardless of their location.
Github		A web-based platform that hosts Git repositories and provides collaborative features for version control and project management. It enables developers to store code, track changes, review and discuss contributions, and manage software development workflows through tools like pull requests, issues, and actions.
Keras		An open-source high-level neural networks API written in Python, designed for easy and fast experimentation with deep learning

Names	Abbreviations	Descriptions
		models. It also represents the extension used by some AI models.
Libraries		A collection of pre-written code that can be used by developers avoiding the need to recreate recurrent functions or functionality.
SDK		A collection of software development tools and libraries that provide a foundation for creating applications for a specific platform or framework.
Union Nationale des Entreprises du Paysage	UNEP	The UNEP is a prominent professional association in France that represents companies and professionals in the landscaping industry.
User pool		A centralized repository or service for managing user accounts and authentication details. It stores information such as usernames, passwords, and other attributes necessary for user identification and access control.

References

- PlantVillage
- Study by IFOP and UNEP
- ResNetV2 documentation
- Keras documentation
- AWS documentation for Flutter
- Dart documentation
- Flutter documentation

- Git Documentation
- GitHub documentation