# Specifications and Conceptions

## Introduction

After presenting the general context of our project, we will dedicate this chapter to formally present our project. We will perform a detailed and complete analysis of different needs related to our application, which will include the following:

- specifying the functional and non-functional requirements of the proposed system

- Identification of the main actors interacting with our application.

-Presentation of global use case diagram

-presentation of different detailed use-case diagrams.

## 2.1 Functional requirements

In the following, we will present the functional requirements of our project.

Functional requirements express the actions that the system must execute in response to user requests. In our case, our application must satisfy the following requirements:

**Registration**

The application must allow users to register in order to access the application

**Authentication**

It allows the user to connect using his email address to benefit from the application’s features.

**Manage Drives**

It allows the user to manage available Drives

**Search files**

It allows the user to search for file in the used drives

**Upload files**

It allows the user to upload files

**Manage Profile:**

It allows the use to consult or update profile

## 2.2 Non-Functional requirements

We identify non-functional needs, which are operational constraints that affect the performance of the application. These are summarized as follows.

* **Scalability and maintenance**: The application must be able to adapt to any change on the implementation side (updating frameworks) to guarantee its evolution and flexibility. In addition,, it must offer a readable, understandable code, and modular.
* **Extensibility**: The system must be scalable and must consider the possibility of its extension by adding new features.
* **Security**: The Application must ensure the security of personal information
* **Performance**: The modules must optimize the treatments and reduce the execution time.
* **Ergonomics**: The application is user-friendly, simple to use, ergonomic, and adapted to the use
* **Reusability**: The components of the system can be reused for the development of different applications.
* **Modularity:** The application is well structured as modules to ensure better readability, a reduction in the risk of error, and the possibility of selective tests.

## 2.3 Identification of Actors

The actor is external to the system. It represents a person or other computer system waiting for one or more services offered by an interface access. It interacts with the system by sending or receiving messages.

The actors that will interact with the system are:

Administrator: This is the person who manages the entire application, which includes managing users and transporters.

User: He is the basic user who benefits from the features of the application.

## 2.4 Use-case diagrams

A use case diagram is a behavior diagram that visualizes the observable interactions between actors and the system under development. The diagram consists of the system, related use cases, and actors.

In this section, we present the needs of our system in a formal manner using unified modeling language (UML) use case diagrams. These features are described in the following diagrams.

### 2.4.1 Global use-case diagram:

The Figure describes, in a general way, the actors of the system as well as the functionalities. A detailed description of the different use cases will follow.

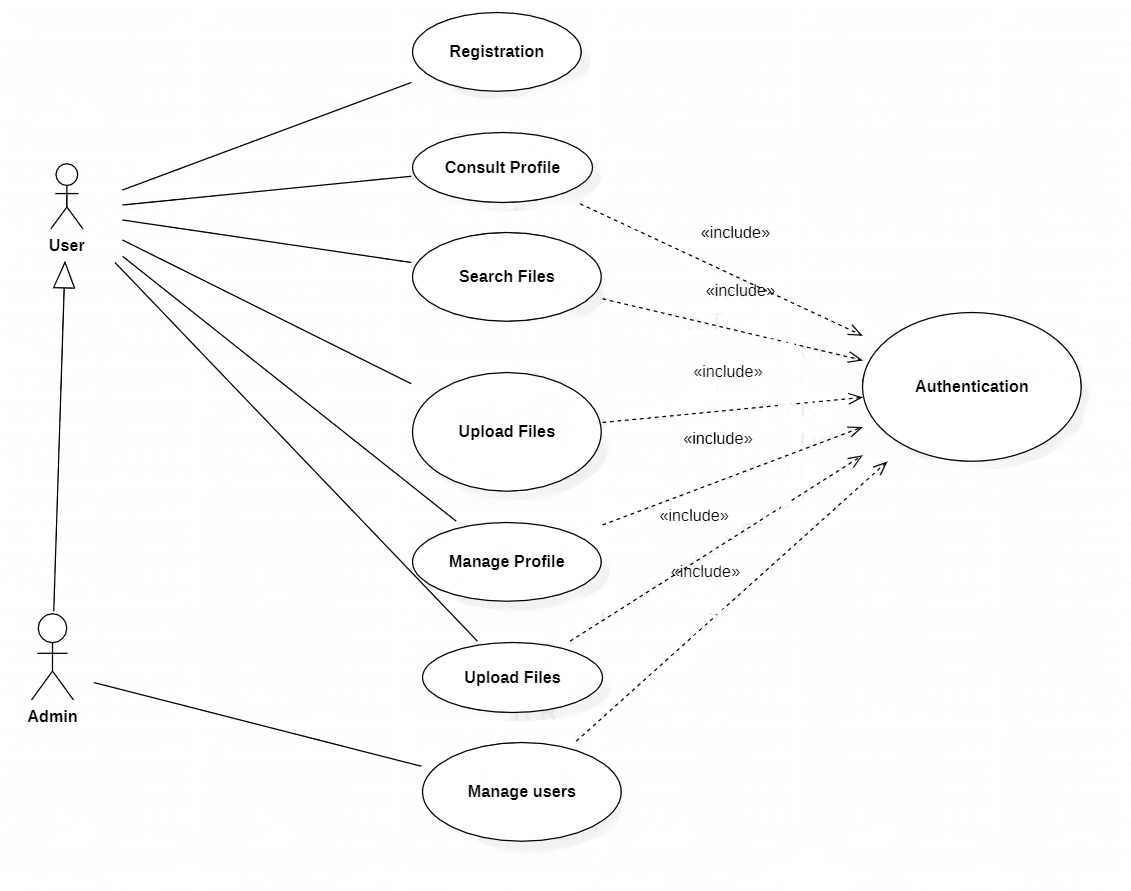


Figure ‎2.1:General use case diagram

### 2.4.2 Authentication use-case diagram:

The figure represents the use-case diagram of authentication. The user will be able to have access to the application with his email and password, as well as restore access to the application through Restore Password Action in case of a forgotten password.

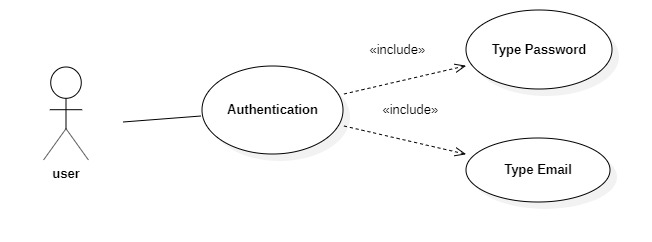


Figure ‎2.2:Authentication use case diagram

Table 2.1 describes in details the authentication use case scenario:

|  |  |
| --- | --- |
| Actor | Admin, User |
| Objective | Get authenticated and access to the application’s features |
| Pre-condition | Registered email and password |
| Post condition | Redirection to Dashboard Page |
| Nominal Scenario | 1-User request the authentication interface  2- the application displays the interface  3-user fills in the necessary and valid fields  4-System checks the data entered and displays the dashboard Page |
| Alternative scenario | 1-User enters incorrect data  2-System displays an error message  3-Resumption of stage 3 of the nominal scenario |

Table ‎2.1:Authentication use case scenario

### 2.4.2 Registration use-case diagram:

The figure represents the use-case diagram of Registration. The user will be able to create an account which he would later use to authenticate.

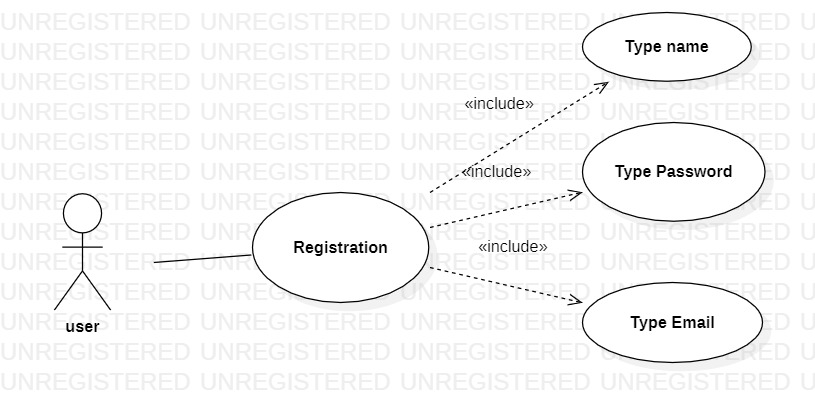


Figure 5:Registration use case diagram

Table 2.1 describes in details the authentication use case scenario:

|  |  |
| --- | --- |
| Actor | Admin, User |
| Objective | Create an account |
| Pre-condition | Typing email and password and name |
| Post condition | Redirection to sign in page |
| Nominal Scenario | 1-User request the sign-up interface  2- the application displays the interface  3-user fills in the necessary and valid fields  4-System checks the data entered and displays the Sign-inPage |
| Alternative scenario | 1-User enters incorrect data  2-System displays an error message  3-Resumption of stage 3 of the nominal scenario |

Table ‎2.1:Authentication use case scenario

## Conclusion

Throughout this chapter, we presented the functional and non-functional needs related to our application, as well as the main functionalities such as workout and chat management, and in the end, we presented the different use-case diagrams and scenarios of those functionalities.