Chapter 2 :Specification needs

1. Introduction

In this chapter, we detail the specification of the functional and non-functional needs in order to complete a high-performance application. Then, we present the actors of this application.

In addition to presenting the management of the work by the chosen methodology and finally we move on to the choice of methodology.

1. Application architecture
2. MVC architecture

We will start by defining the Model–view–controller, MVC is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements.

* Model: The central component of the pattern. It is the application's dynamic data

structure, independent of the user interface. It directly manages the data, logic, and rules of the application.

* View: It defines the user interface such as a chart, diagram, or table. This part focuses on the display of information to the end-user.

* Controller: it orchestrates the flow of data between the view and the model. It’s the logical point which allows you to manage events, handle various conditions and ensure synchronization.

1. Software architecture

for our system, we will opt for an architecture illustrated in Figure 3.

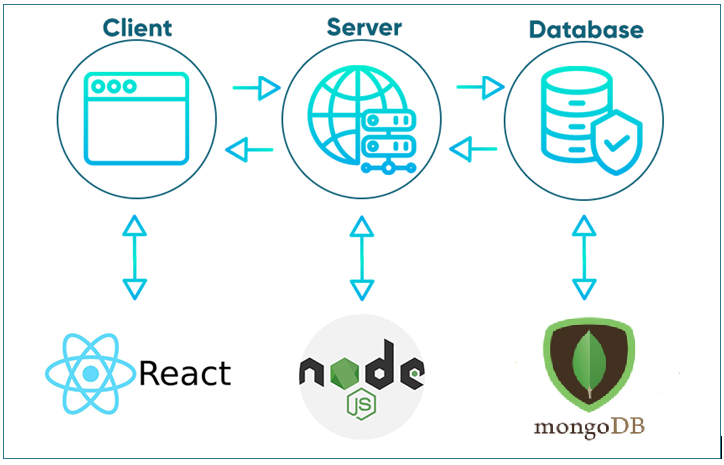


Figure 2 :Application software architecture

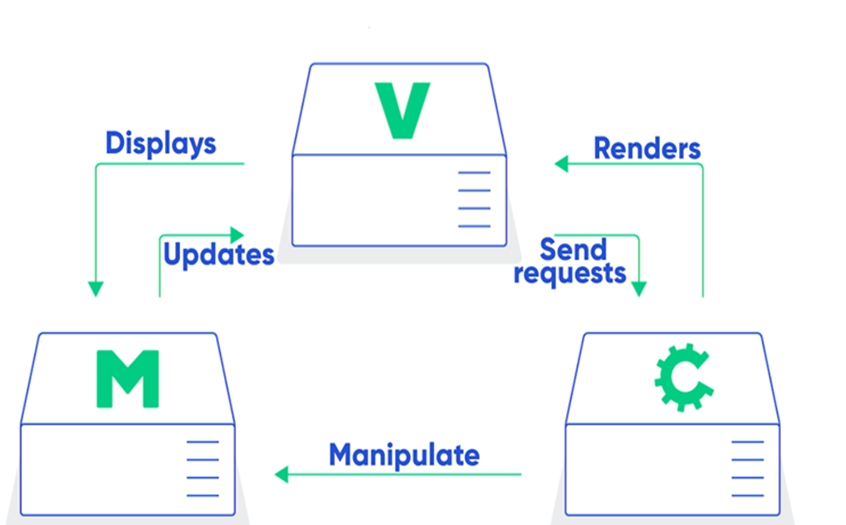


Figure 3: MVC structure

Figure 4 : MVC architecture

**Client**: the graphical interfaces of the application and displaying data from the database.

**Server**:  It contains all the application's logical processes such as the launching the server

functions.

**Database**: is responsible for storing, querying and persistence the data.

The actors are all the individuals who will intervene in this application. We will identify the list of actors in our system as presented in the following table down below:

Table 3 :Actors roles

|  |  |
| --- | --- |
| **Actors** | **Role** |
| Admin | * Manage clients * Create estimation * Manage orders * Update information’s |
| Client | * Order product * Demand a service * Update account * Manage estimation * Create account |
| Visitor | * View products      * Sign up |

1. Specification of functional needs

The functional needs represent the actions that the system must perform.

* Admin:

* The admin should be able to manage clients, orders, and admin dashboard.

* Client:

* Client should be able to Make an order.

* Client should be able to Demand an Estimation.

* Client will be able to view their orders.

* Visitor:

* View the website.

* Sign up for new account.

1. Specification of non-functional needs

These needs identify the internal and external constraints of the system.

* ***Security***: Needs to establish the connection securely, you must have an authentication interfacethat allows different users to connect to view their unique profiles safely.

* ***Usability design***: The site should be easy enough to use and have a self-explanatory feeling to it. It must present a logical sequencebetween the interfaces and a set of links to ensure smooth and fast navigation, an understandable, visible, and readable text.

* ***Availability***: Whenever any user willing to consult the site, it must be available and fully functional.

* ***Response time***: The response time should be as short and feels responsive to the user.

1. SCRUM

Scrum is an agile framework for managing and organizing complex projects. It is commonly used in software development but can be applied to various other fields as well. Scrum provides a flexible and iterative approach to project management, focusing on collaboration, adaptability, and delivering high-quality products.

In Scrum, projects are divided into small, manageable units called "sprints," typically lasting 1-4 weeks. During each sprint, a cross-functional team works on a set of prioritized tasks, known as the "backlog," which is a list of features, enhancements, or bug fixes. The team collaborates closely, with frequent communication and coordination.

The Scrum team consists of three key roles:

Product Owner: Represents the stakeholders and defines the project's vision, priorities, and requirements. They manage the product backlog and ensure that the team is working on the most valuable tasks.

Scrum Master: Facilitates the Scrum process and ensures that the team adheres to Scrum principles. They remove any obstacles or impediments that might hinder the team's progress and help create a productive and collaborative environment.

Development Team: Self-organizing and cross-functional, the development team is responsible for delivering the project's increments during each sprint. They estimate and select the tasks they can complete within the sprint and work together to accomplish them.

Scrum emphasizes continuous improvement through regular feedback and review. At the end of each sprint, the team conducts a sprint review to demonstrate the completed work to stakeholders and gather feedback. They also hold a sprint retrospective to reflect on the process and identify areas for improvement in the next sprint.

Strengths:

•Flexibility and adaptiveness

•Scrum ensures effective use of time and money

•Creativity and Innovation

•Lower Costs

•Ensures Continuous Feedback

•The team gets clear visibility through scrum meetings

•Timely Predictions

Weaknesses: •Requires a Team Environment

•Not appropriate for Plan driven approach projects

•The framework can be successful only with experienced team members

•Not Appropriate for Large and Complex Projects

•Requires Transformation at Organization Level

The main benefits of Scrum include increased transparency, adaptability to changing requirements, faster delivery of valuable increments, and improved collaboration within the team. It promotes a customer-centric approach and encourages iterative development, allowing for early and frequent delivery of working software or products.

6.Identification of actors

* Admin:

* The admin should be able to manage clients, orders, and admin dashboard.

* Client:

* Client should be able to Make an order.

* Client should be able to Demand an Estimation.
* Client should be able to Accept or Refuse an Estimation.
* Client will be able to view their orders.

* Visitor:

* View the website.

* Sign up for new account.

7.Definition of UML

Unified Modeling Language (UML) is a pictogram-based graphic modeling language designed to provide a standardized method for visualizing system design. It is commonly used in software development and object-oriented design

8.The global case diagram.

* + 1. Use Case diagrame

The class diagram represents the classes that make up the system and the associations between them. It expresses in a general way the static structure of a system.

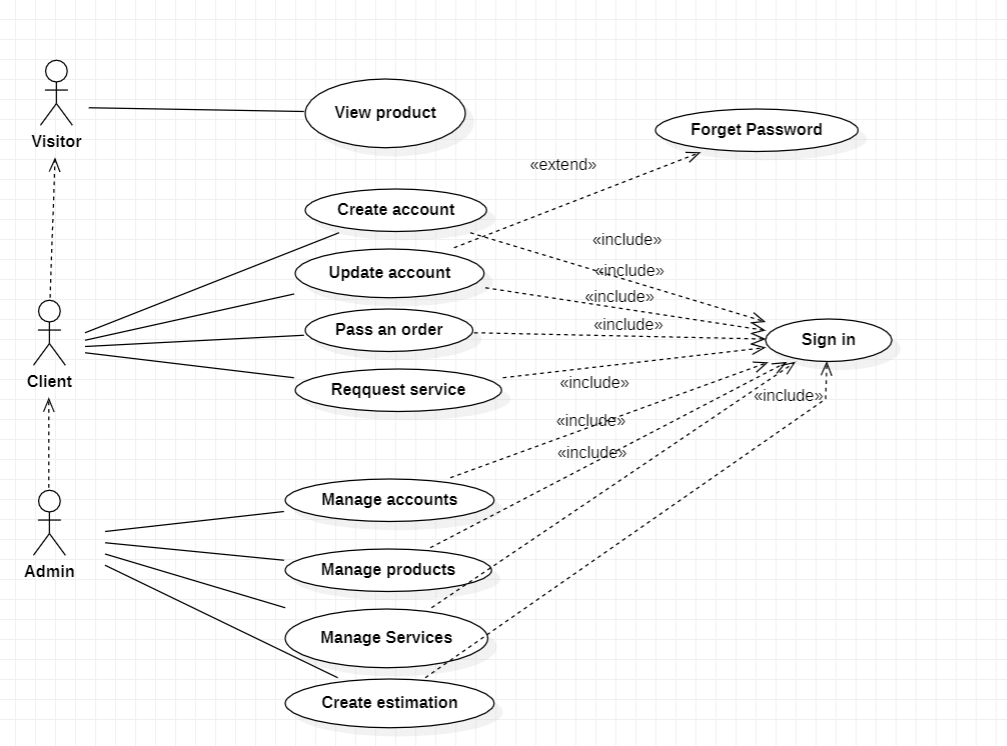


Figure 5 :use case diagrame

* + 1. Global case diagram

The class diagram represents the classes that make up the system and the associations between them. It expresses in a general way the static structure of a system.

.

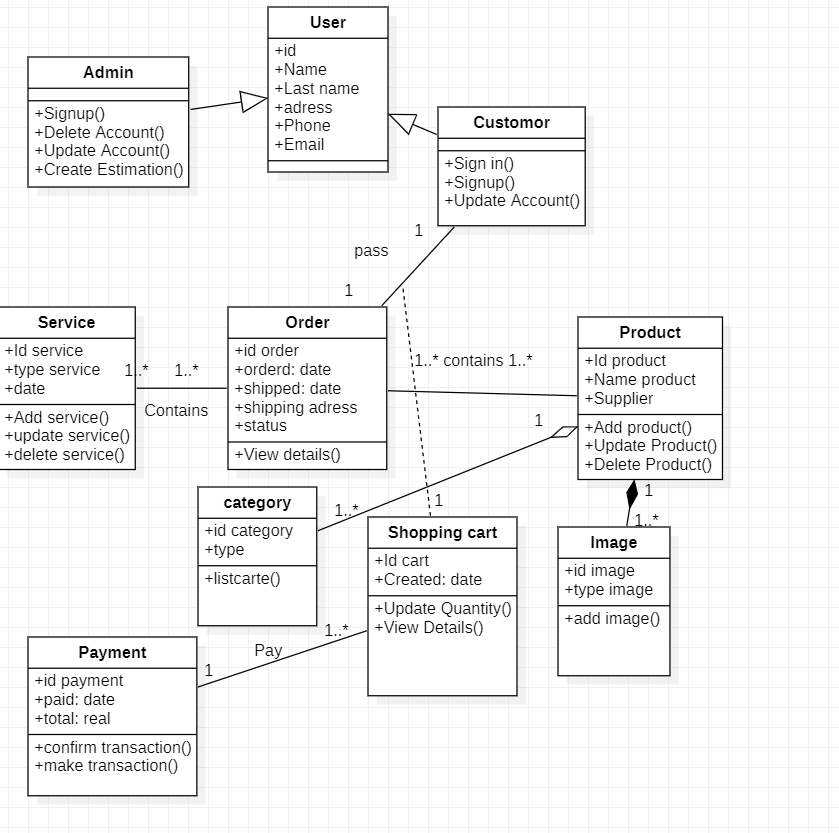


Figure 6:Global Case Diagram

9. Conclusion

In this chapter we have detailed the different conceptual views of the applications to be realized through the necessary UML models and The next chapter will highlight the course, the different sprints and level of development.

# **Chapter 3**

# **Release 1**

Chapter 3 : Release 1

1. Introduction

In this chapter, we will work on detailing every sprint by showing the use case diagram, global case diagram and sequence diagram also the refinement, backlog sprint and finally the realisation.

2.Sprint 0: Technical choice

## 2.1 Material environment

To complete this project and do the necessary testing, a ASUS laptop has been made available to us with the following specifications:

Table 4 :Hardware configuration

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  |  |
| **Manufacture company** |  | MSI |
|  |  |  |
| **Processor** |  | core i7 10th Gen |
|  |  |  |
| **RAM** |  | 16 Go |
|  |  |  |
| **Disque** |  | 1To |
|  |  |  |
| **Graphic card** |  | Nvidia GTX 1650 |
|  |  |  |
| **Operation System** |  | Windows 11 |
|  |  |  |

## 

## 2.2 Software environment

On this part we reveal the various tools and technologies used throughout the making of this project.

* **UML modelling tool «Visual Paradigm Online»:** a UML modelling tool online solution.
* **Code editor «Visual Studio Code»:** Visual Studio Code is a source-code editor made by Microsoft, combines the simplicity of acode editor with what developers need for their developing cycles.
* **Postman:** is an API client that makes it easy for developers to create, share, test and document APIs.

# 2.3. Choice of programming stack

After a throughout study of the different technologies we have come to conclude that what best fits this project is a combination of Electron.js, React, node.js and MongoDB.

* **React**: React is an open-source JavaScript library for building user interfaces. React can be used as a base in the development of single-page or mobile applications.

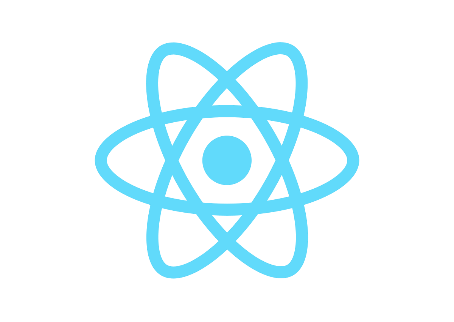


Figure 7 :logo "ReactJS"

**→** React.js is one of the best JavaScript libraries that's why we found many big platforms made with it. For example:

* **Node.js:** Node.js is an open-source, cross-platform, JavaScript runtime environmentthat executes JavaScript code outside a web browser.

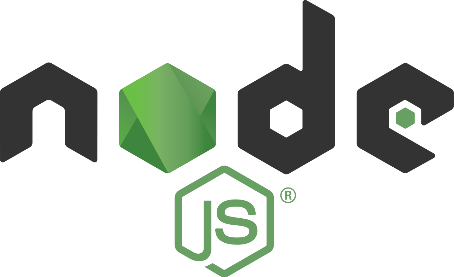


Figure 8:logo "NodeJS"

* **MongoDB:** is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, uses JSON-like documents with optional schemas.



Figure 9 : MongoDB

* **ElectronJs:** Electron is an open-source software framework developed andmaintained by GitHub. It allows for the development of desktop GUI applications using web technologies: it combines the Chromium rendering engine and the Node.js runtime.



Figure 10:logo "ElectronJS"

* **ExpressJs:** is a back-end web application framework for Node.js It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.



Figure 11:ExpressJs logo

**[EmailJS:](https://www.emailjs.com/)**

EmailJS helps to send emails using client-side technologies only. No server is required – just connect EmailJS to one of the supported email services, create an email template, and use one of our SDK libraries to trigger an email.



Figure 12 : Emailjs Logo

3.Sprint 1: Sign in:

This sprint aims to develop the first part of our project which is Authentication.

✓ Backlog de sprint1

.

Table 5:Backlog of first sprint

|  |  |
| --- | --- |
| Client | Priority |
| Sign in | High |

Sprint 1 Detailed Use Case Diagram

Thereafter, we present the phase of analysis of which we describe the case diagram.

then the textual description of each of them.

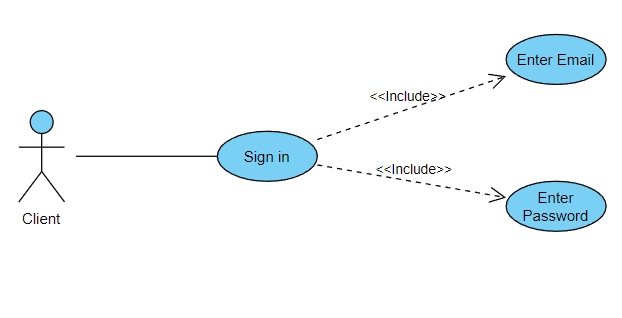


Figure 13 :use case diagram of client

## Refinement of use case « Sign in»

|  |  |
| --- | --- |
| Use case | Sign in |
| Actor | Client |
| Precondition | Email and Password must be valid |
| Condition | Client is logged in |
| Scenario | [Begin]  1.The user requests the authentication interface  2.The system displays the authentication interface.  3.The user enters his login and password  4.The user confirms by clicking on the “Login” Button  5.The System Checks the data Entered.  6.The system displays the user-specific home interface to the user  [End] |
| Alternative |  |

The first scenario when the application is launched. According to Figure , the user should enter their login data to have the access of the application features. The system checks the availability of the login and password, then gives access to the functionality of the application.

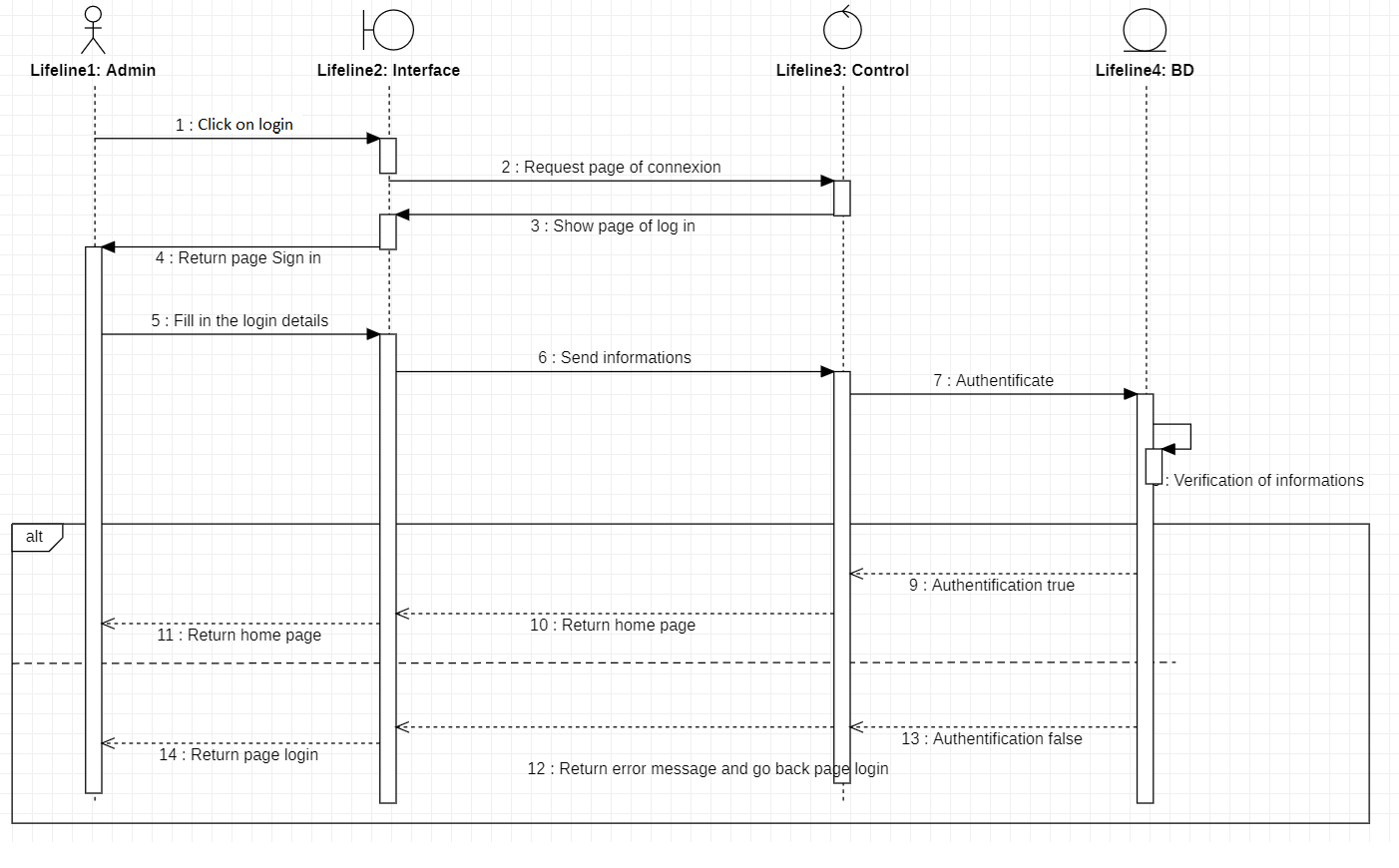


Figure 14 :sequence diagram of sign in

✓ Class diagram: "Authentication"

Figure presents the class diagram modelling the “Sign in” module :

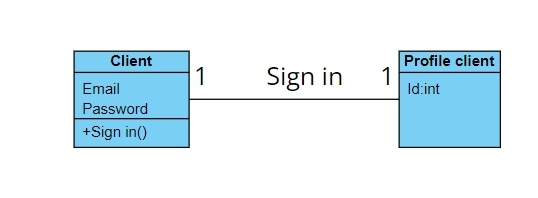


Figure 15:class diagram of sign in

2.3 Realization

In order to show the results of this sprint, we expose some interfaces of our application through different screen prints made

✓ The interfaces made in this sprint

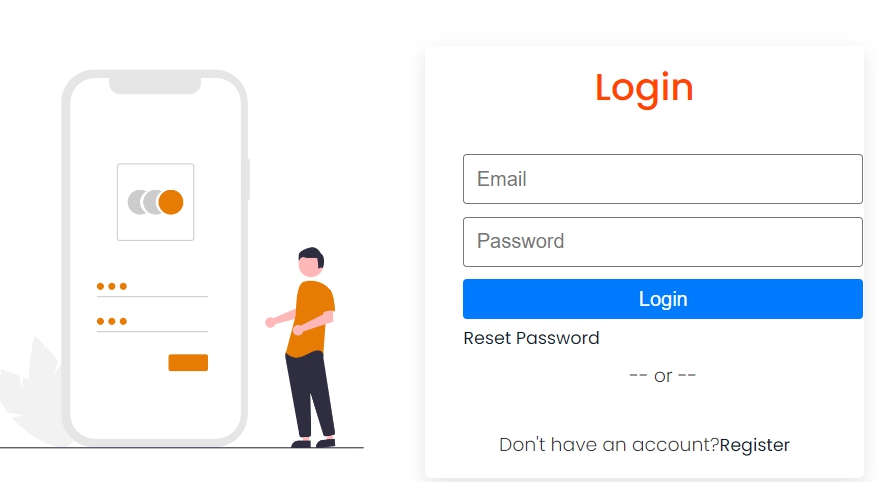


Figure 16:Login interface

Figure shows our different authorization keys to protect our different

app features.

Our application is controlled by two keys:

1. A global key for the "password" application: This key corresponds to the means of

security of our global application.

2. A unique "email" key: This key corresponds to a single user. Each

participant has its own key.

✓ The interfaces made in this sprint

This part is devoted to the exhibition of the completed work through screenshots of

different interfaces developed during this sprint.

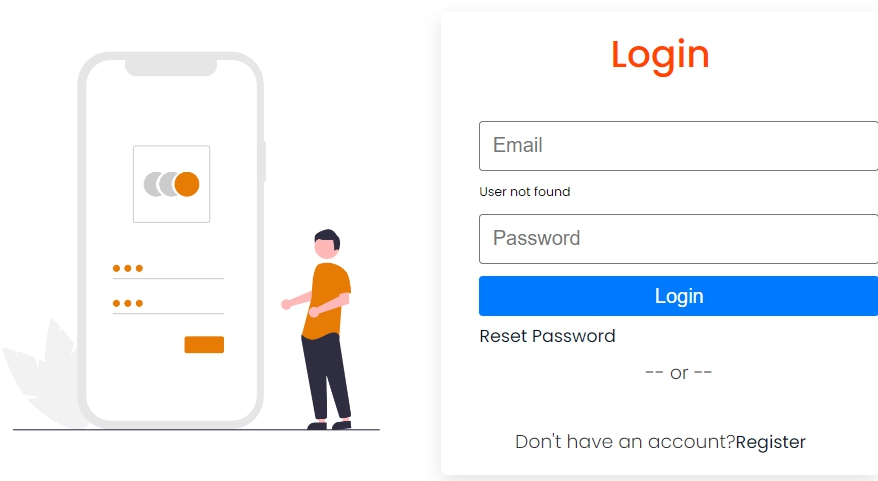


Figure 17:First condition of login on email

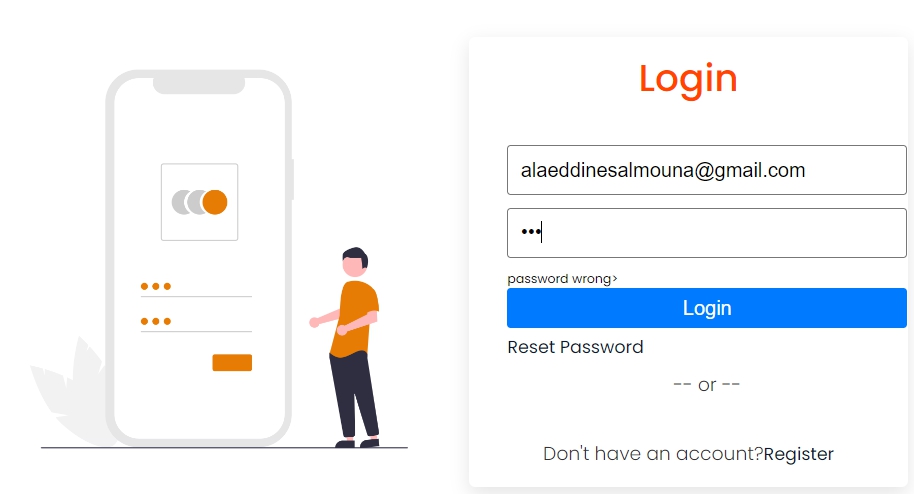


Figure 18:second condition of login wrong password

Those are the 2 figures if the details entered false .

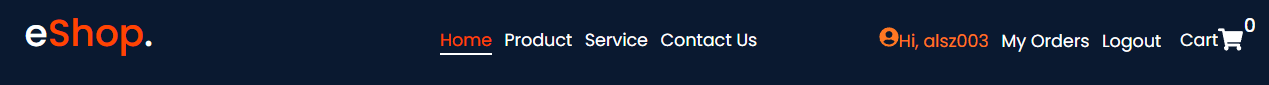


Figure 19:Home interface after login

If the details entered true here is the home page where the client can see products,service contact the company or see his orders .

4.Sprint 2: Sign up client

This sprint aims to develop the second part of our project which is Signup.

✓ Backlog de sprint1

Table 6:Backlog du premier Sprint.

|  |  |
| --- | --- |
| Client | Priority |
| Sign up | High |

Sprint 1 Detailed Use Case Diagram

Thereafter, we present the phase of analysis of which we describe the case diagram

then the textual description of each of them.

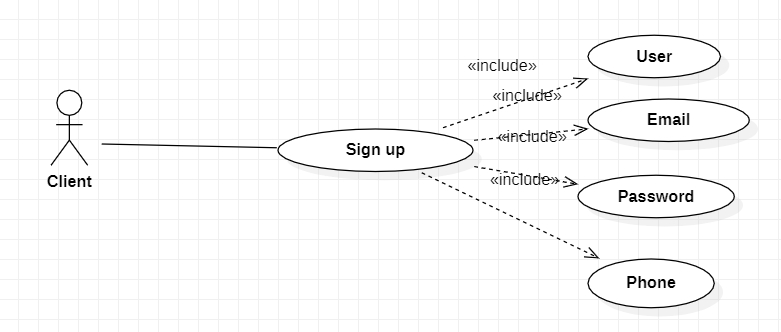


Figure 20 : Sign up use case diagram

**Refinement of use case «Sign up».**

|  |  |
| --- | --- |
| Use case | Authentication |
| Actor | Visitor |
| Precondition | User, Email,Password,Phone must respect conditions |
| Condition | The Visitor visit the website and want to sign up |
| Scenario | [Begin]  1)The user fills in the information’s  2)The user press Sign up Button  2)The System checks in the conditions  3)If everything is okay the sign up went successfully  4)The System Displays the Home Interface  [End] |
| Alternative | The System shows notification message . |

The first scenario when the application is launched. According to Figure , the user when he want to signup he need to enter their signup data to register a new account. The system checks the availability of the user,email,password and phone, then gives access to the functionality of the application.

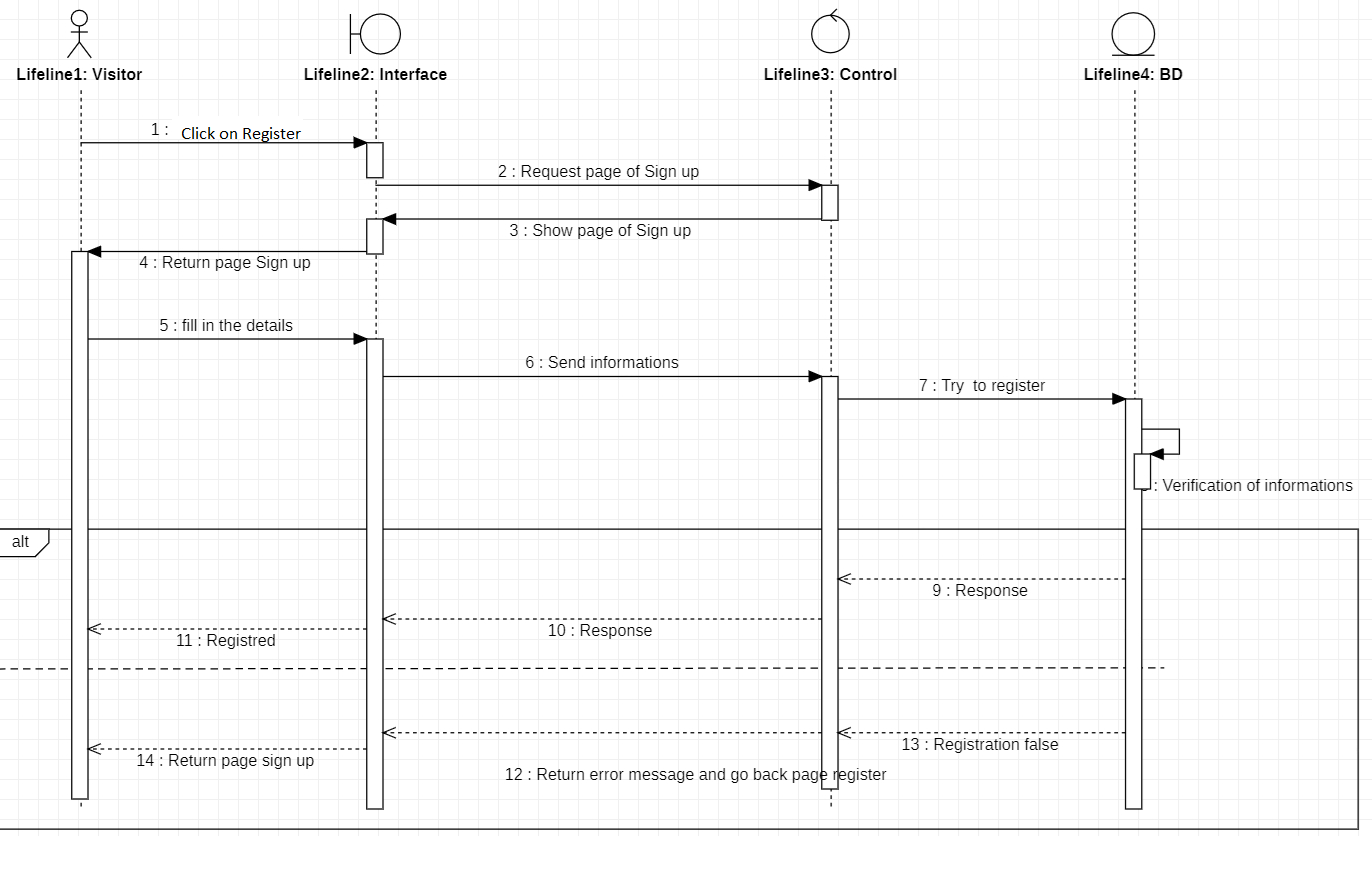


Figure 21:sequence diagram of sign up

✓ Class diagram: "Sign up"

Figure presents the class diagram modelling the “Sign up” module :

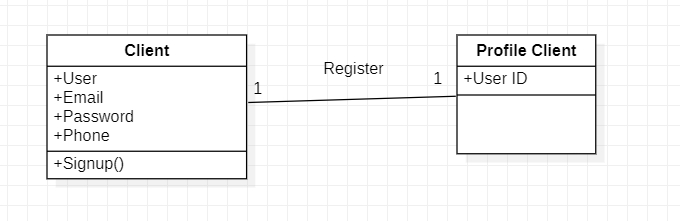


Figure 22:Global case diagram of sign up

2.3 Realization

In order to show the results of this sprint, we expose some interfaces of our application through different screen prints made

✓ The interfaces made in this sprint

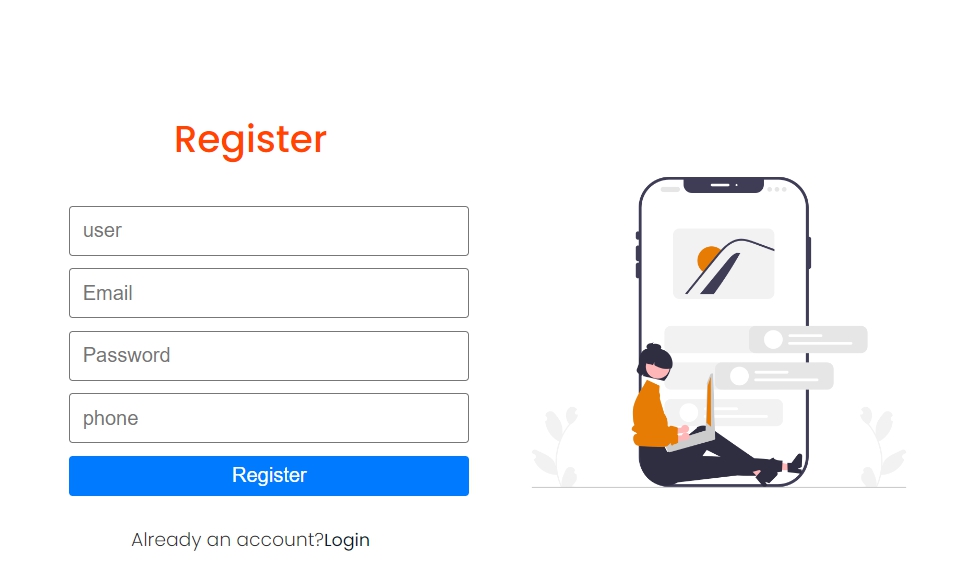


Figure 23:Sign up interface

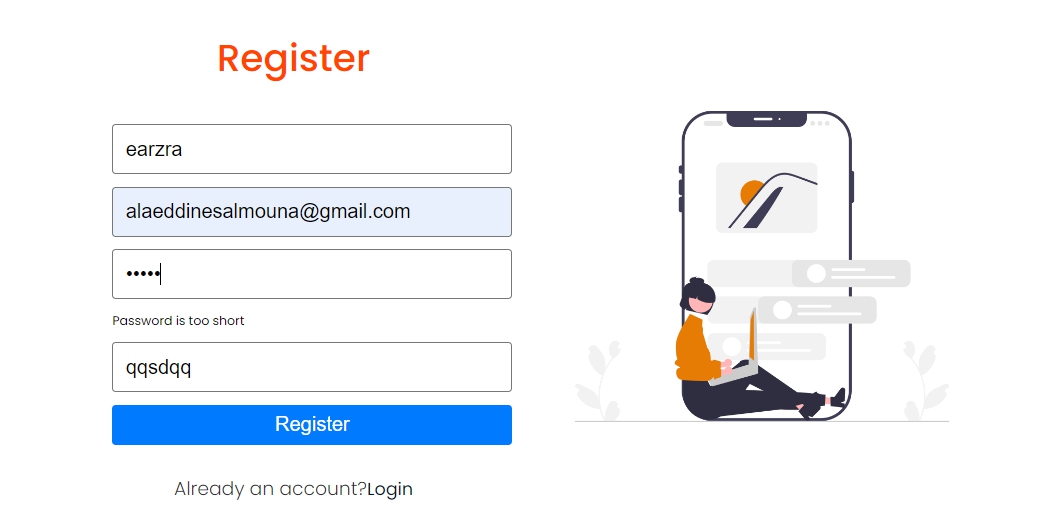


Figure 24:condition on the password

When password is to short it pop up an error “password is too short”

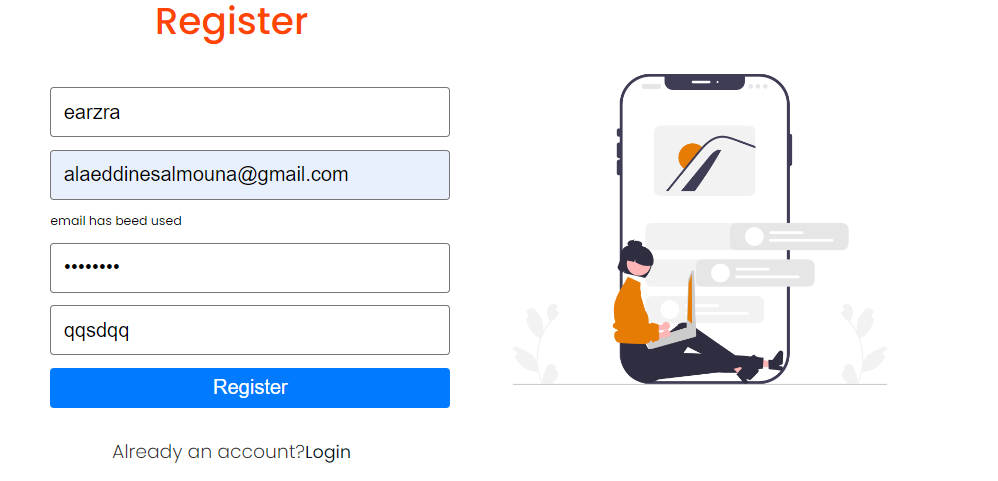


Figure 25:Condition on the email

When email is used it declares an error

4.Conclusion

In this chapter we worked on detailling the technical choice and sprint of sign in and sign up with detailed sequence ,use case and global diagram and refinement also we showed the interfaces of those sprints .

# **Chapter 4**

# **Release 2**

Chapter 4 :release 2

1.Introduction:

In this chapter we are going to detail the sprint 3 which is pass order, sprint 4 update account and finally sprint 5 request service

1.Sprint 3: Pass order:

This sprint aims to let the client make an order.

✓ Backlog de sprint3

Table 7 :Backlog of the third sprint

|  |  |
| --- | --- |
| Client | Priority |
| Place order | Medium |

Sprint 3 Detailed Use Case Diagram

Thereafter, we present the phase of analysis of which we describe the case diagram

then the textual description of each of them.

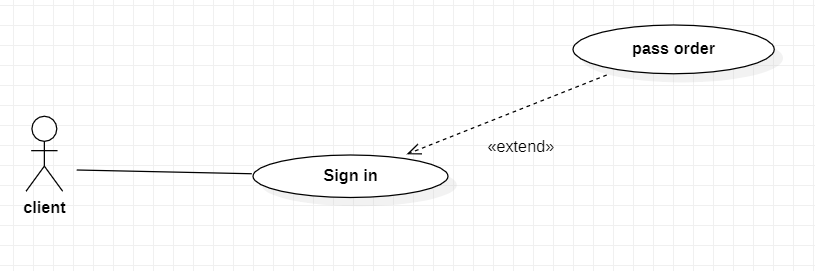


Figure 26:Pass order use case diagram

## Refinement of use case « Pass order»

|  |  |
| --- | --- |
| Use case | Authentication |
| Actor | Client |
| Precondition | Email and Password Must be Valid |
| Condition | Client is logged in |
| Scenario | [Begin]  1.The client access to the products tab .  2.the system displays the products interface.  3.the client chose the product.  4.the client add to the cart .  5.the client make the checkout by filling his shipping address .  6.the client confirms the order . |
| Alternative | The system displays and error message if the login or password wrong or and a message if the shipping address isn’t filled correctly . |

The first scenario when the application is launched. According to Figure 28, the user should enter their login data to have the access of the application features. The system checks the availability of the login and password, then gives access to the functionality of the application then the client chose the product add it to the cart then select quantity and then press Checkout after that they fill out the shipping address and press procced to checkout.

✓ Class diagram: "Place order"

Figure presents the class diagram modelling the “Place order” module:

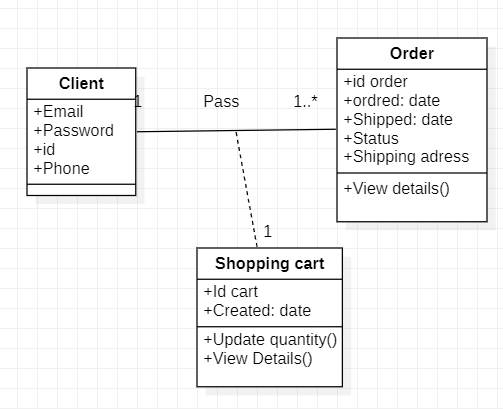


Figure 27:Global case diagram of pass order

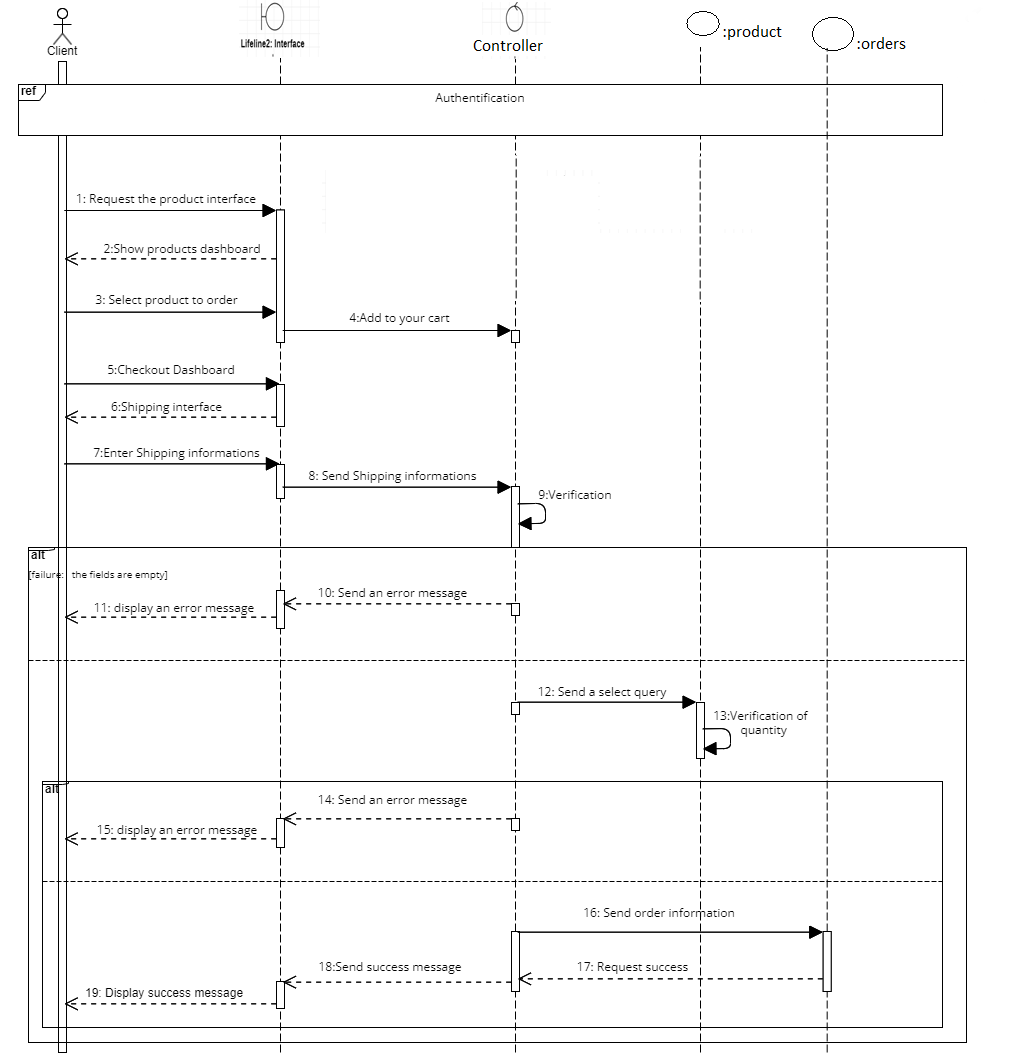


Figure 28:Pass order

2.3 Realization

In order to show the results of this sprint, we expose some interfaces of our application through different screen prints made

✓ The interfaces made in this sprint

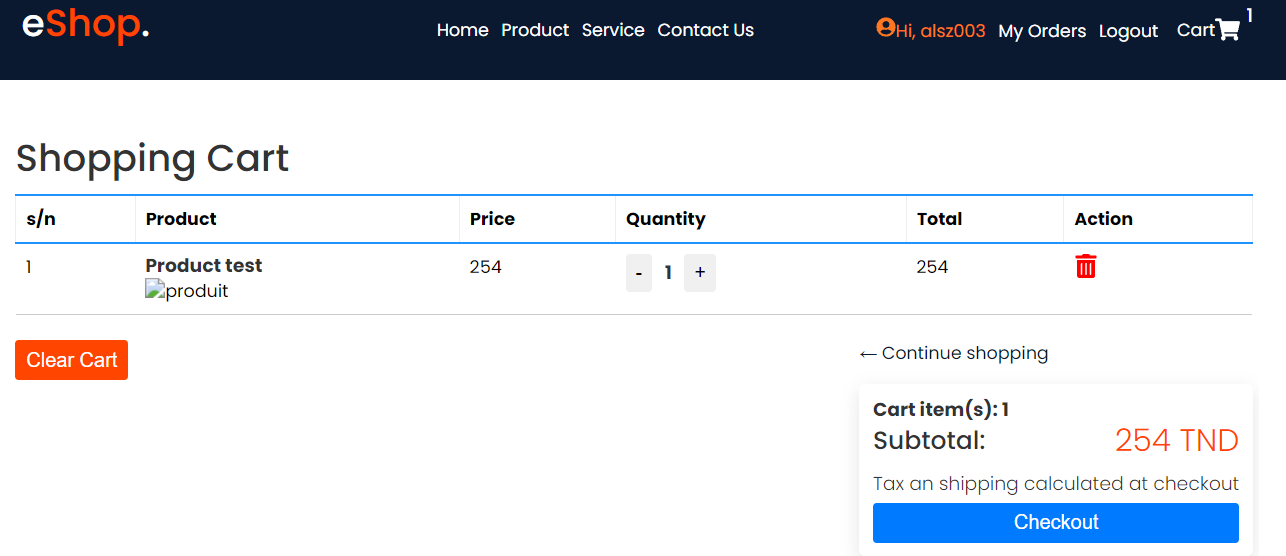


Figure 29:Add product to my cart

In this figure we see how the user has added his product and going to procced the checkout .

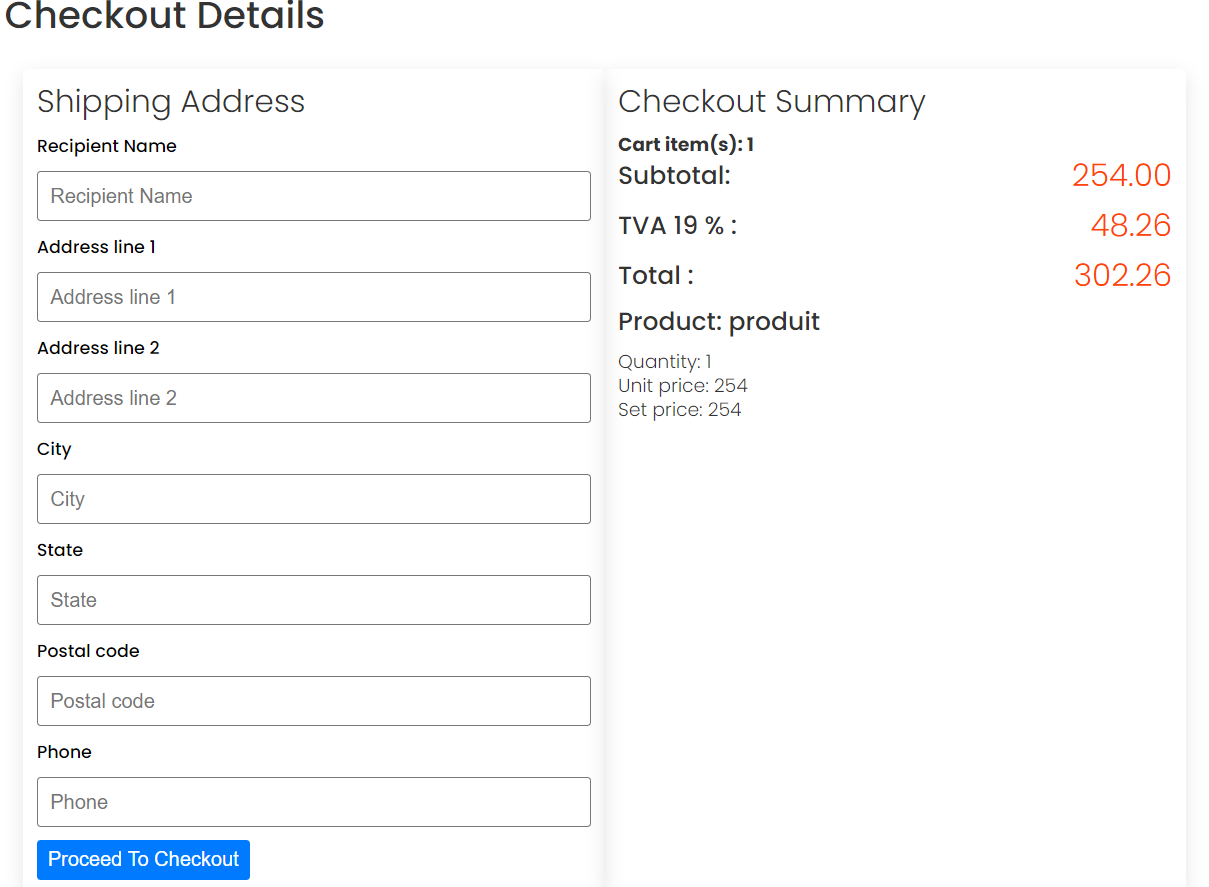


Figure 30:Shipping adress

In this figure we see client filling his shipping address

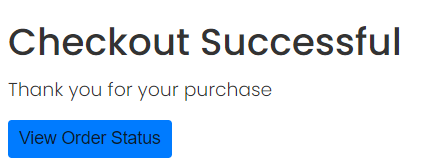


Figure 31:Checkout Successful

Here we see the order successfully passed .

Sprint 4: Update account

This sprint aims to update the client profile .

✓ Backlog de sprint4

Table 8 :Backlog of the third sprint

|  |  |
| --- | --- |
| Client | Priority |
| Update account | Medium |

Sprint 4 Detailed Use Case Diagram

Thereafter, we present the phase of analysis of which we describe the case diagram

then the textual description of each of them.

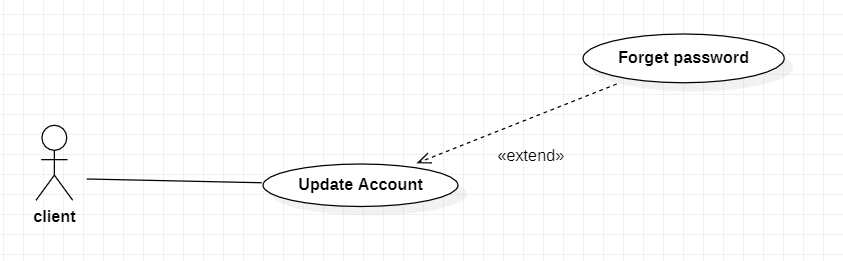


Figure 32:Update account

## Refinement of use case « Pass order»

Table 9:Update account

|  |  |
| --- | --- |
| Use case | Authentication |
| Actor | Client |
| Precondition | Email and Password Must be Valid |
| Condition | Client is logged in |
| Scenario | [Begin]  1.The client access to my profile dashboard. .  2.the system displays My profile interface.  3.the client click on edit...  4.the client change the information’s.  6.the client confirms the update. |
| Alternative | The system displays and error message if the login or password wrong or and a message if the shipping address isn’t filled correctly . |

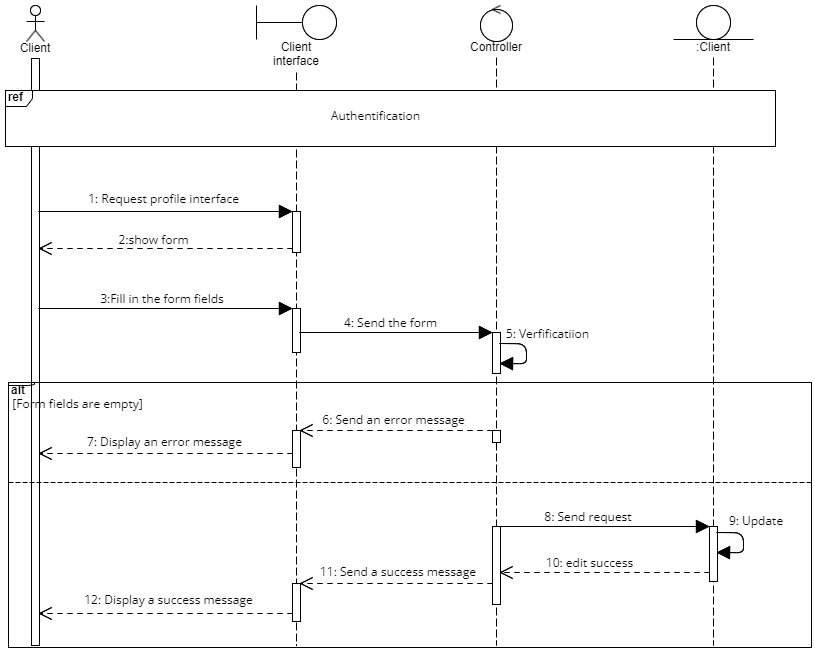


Figure 33:update account

✓ The interfaces made in this sprint

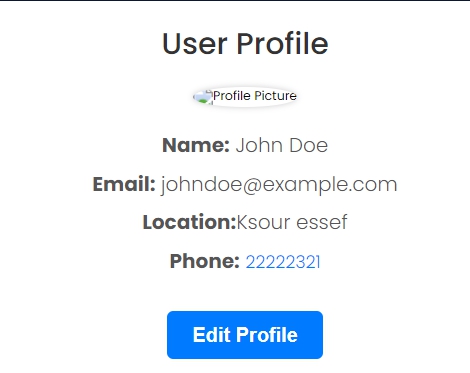


Figure 34:edit profile interface

Sprint 5: Request a service:

This sprint aims to let the client to request a service .

✓ Backlog de sprint 5

|  |  |
| --- | --- |
| Client | Priority |
| Request service | Medium |

Figure 35:Backlog of sprint 5 request a service

Sprint 5 Detailed Use Case Diagram

Thereafter, we present the phase of analysis of which we describe the case diagram

then the textual description of each of them.

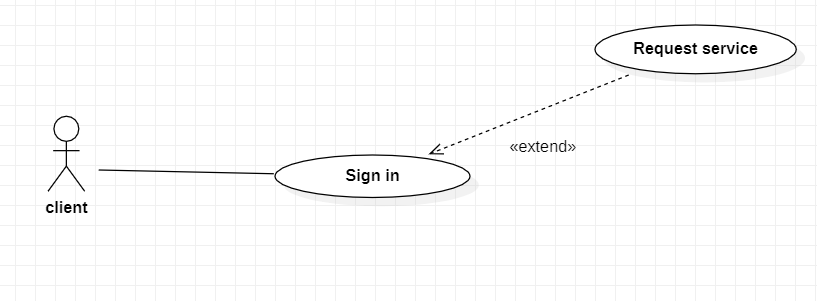


Figure 36:use case of request a service

**Refinement of use case «Request a service».**

|  |  |
| --- | --- |
| Use case | Authentication |
| Actor | Client |
| Precondition | Email and password must be valid |
| Condition | Client is logged in |
| Scenario | [Begin]  1.The client access to the service tab .  2.the system displays the products interface.  3.the client chose the type of service.  4.the client fill the information’s requested .  5. the client confirms the order  [End] |
| Alternative | The System displays and error if login details are incorrect |

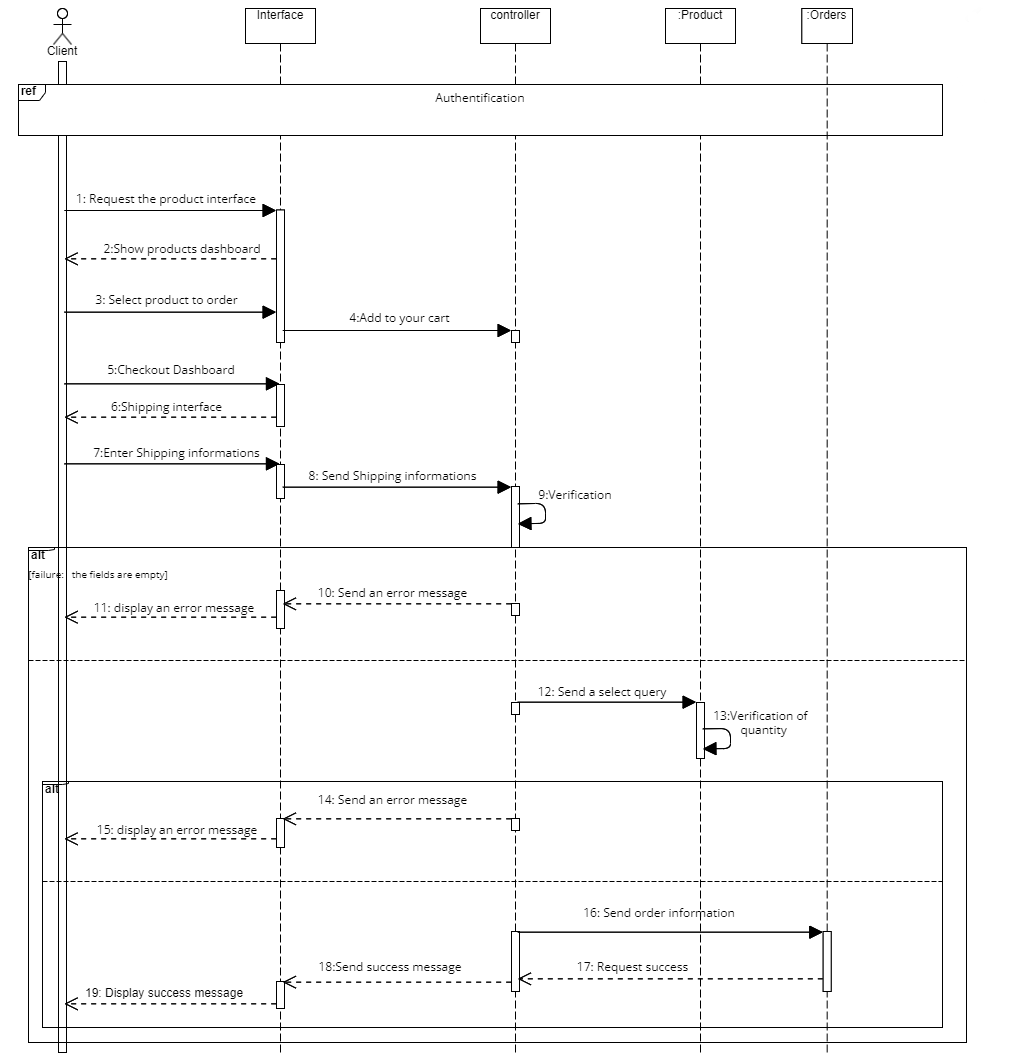


Figure 37:use case of sequence diagram of requesting a service

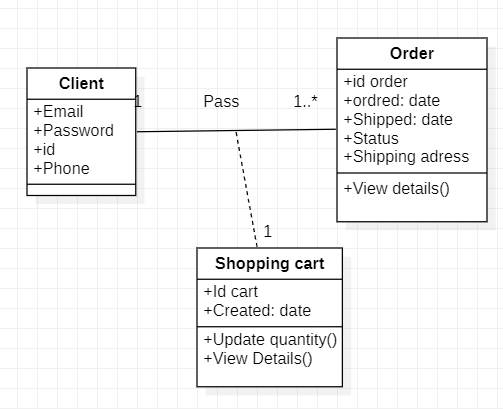


Figure 38:Global case of request a service

2.3 Realization

In order to show the results of this sprint, we expose some interfaces of our application through different screen prints made

✓ The interfaces made in this sprint

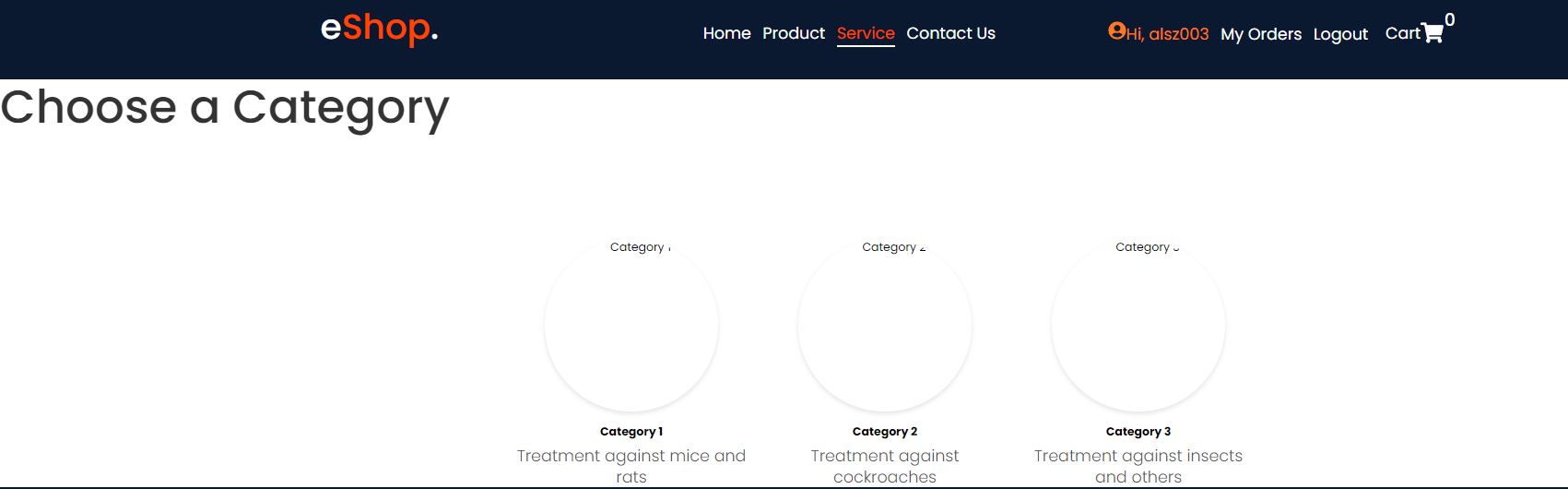


Figure 39:Services types

Here the client will chose which service he needs .

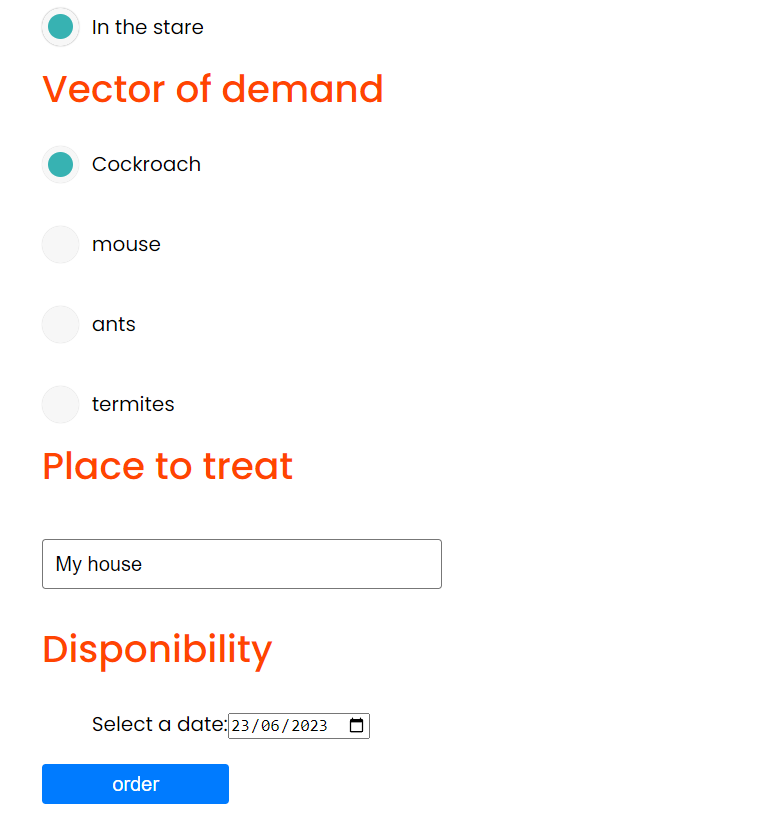


Figure 40:Informations to fill in a service.

Here the client will chose the options provided to make to the company easier to understand what is the needs .



Figure 41:Service Has been sent succefully.

Here we can see that the service is now sent and waiting for review.

# 

# **Chapter 5**

# **Release 3**

# Chapter 5: Release 3

1. Introduction

In this chapter we will be talking about sprint 7: manage product ,service 8: manage services ,sprint 9: manage orders and finally manage client then we will find a conclusion .

**1.1.Admin manage Product: Sprint 7:Manage product**

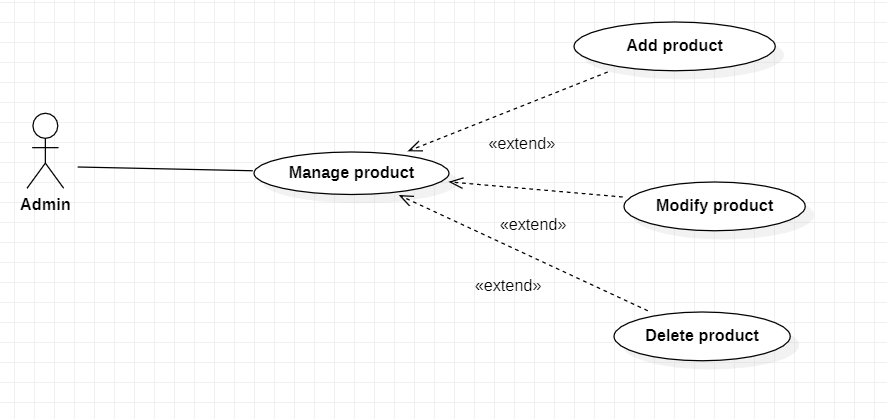


Figure 42:Manage products use case diagram

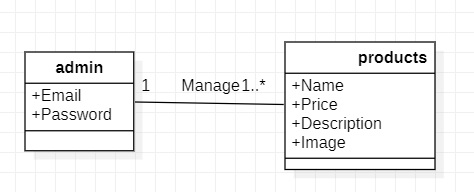


Figure 43:Manage Products by admin

**1.2Refinement of use case «Manage Products: Add product»:**

Table 10:Refinement of use case manage products.

|  |  |
| --- | --- |
| Use case | Authentication |
| Actor | Admin |
| Precondition | Admin must be logged in |
| Condition | Admin is logged in |
| Scenario | [Begin]  1.The admin access to the My Products tab.  2.the system displays Product interface.  3.the Admin chose the chose Add product.  4.the Admin click "confirm".  [End] |
| Alternative | The system displays an error message if case existing product. |

The figure 41 represents the sequence diagram for the use case « Manage Product Add product » of the admin.

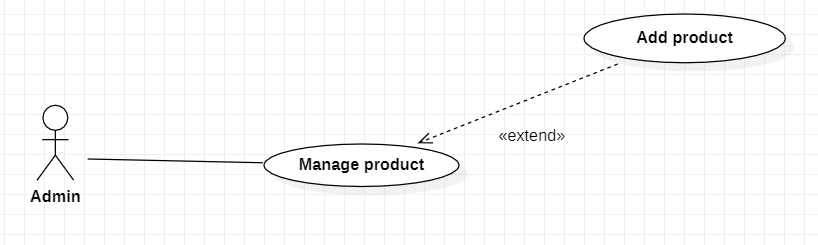


Figure 44:add product use case

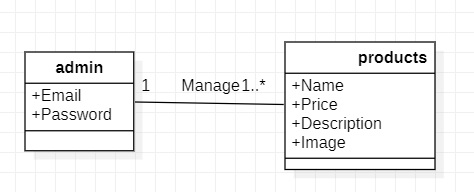


Figure 45:Global case diagram of add product

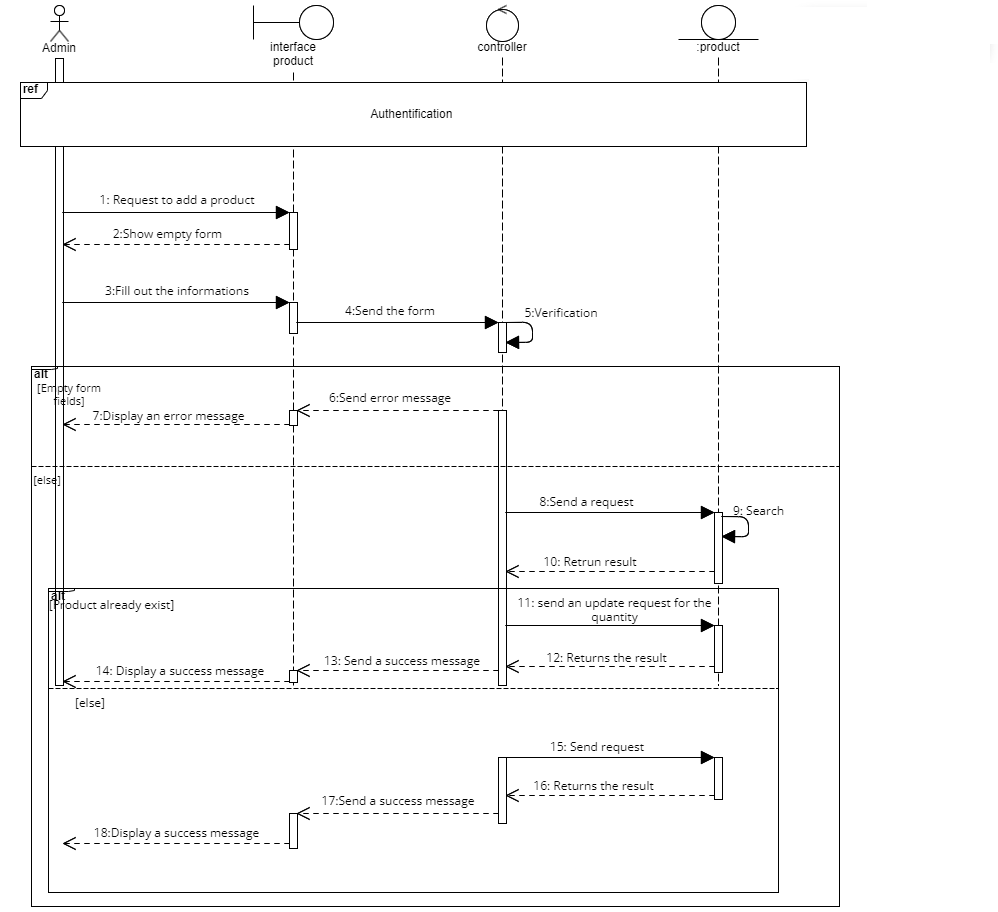


Figure 46:add product sequence diagram

2: Realisation :

**1.3.Admin manage Product: Modify Product Sprint 7**

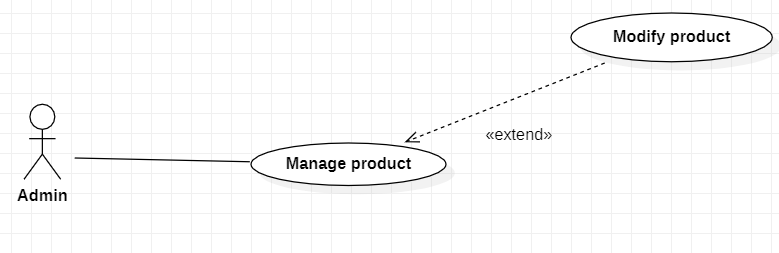


Figure 47:Manage products use case diagram

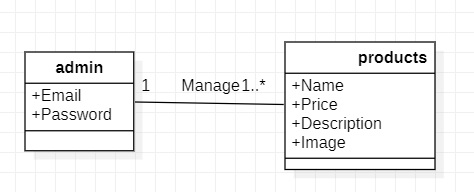


Figure 48:Manage Products by admin

* 1. **Refinement of use case «Manage Products: Modify product»:**

Table 11:Refinement of use case manage products.

|  |  |
| --- | --- |
| Use case | Authentication |
| Actor | Admin |
| Precondition | Admin must be logged in |
| Condition | Admin is logged in |
| Scenario | [Begin]  1.The admin access to the My Products tab.  2.the system displays Product interface.  3.the Admin chose the chose Modify product.  4.Fill in the needed information’s .  5..the Admin click "confirm".  [End] |
| Alternative | The system displays an error message if case existing product. |

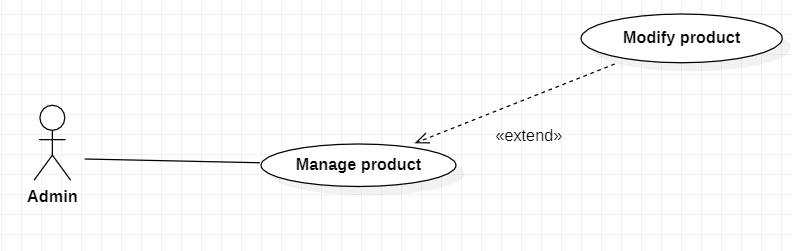


Figure 49:Modify product use case

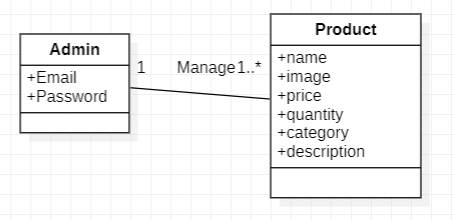


Figure 50:Global case diagram of manage product

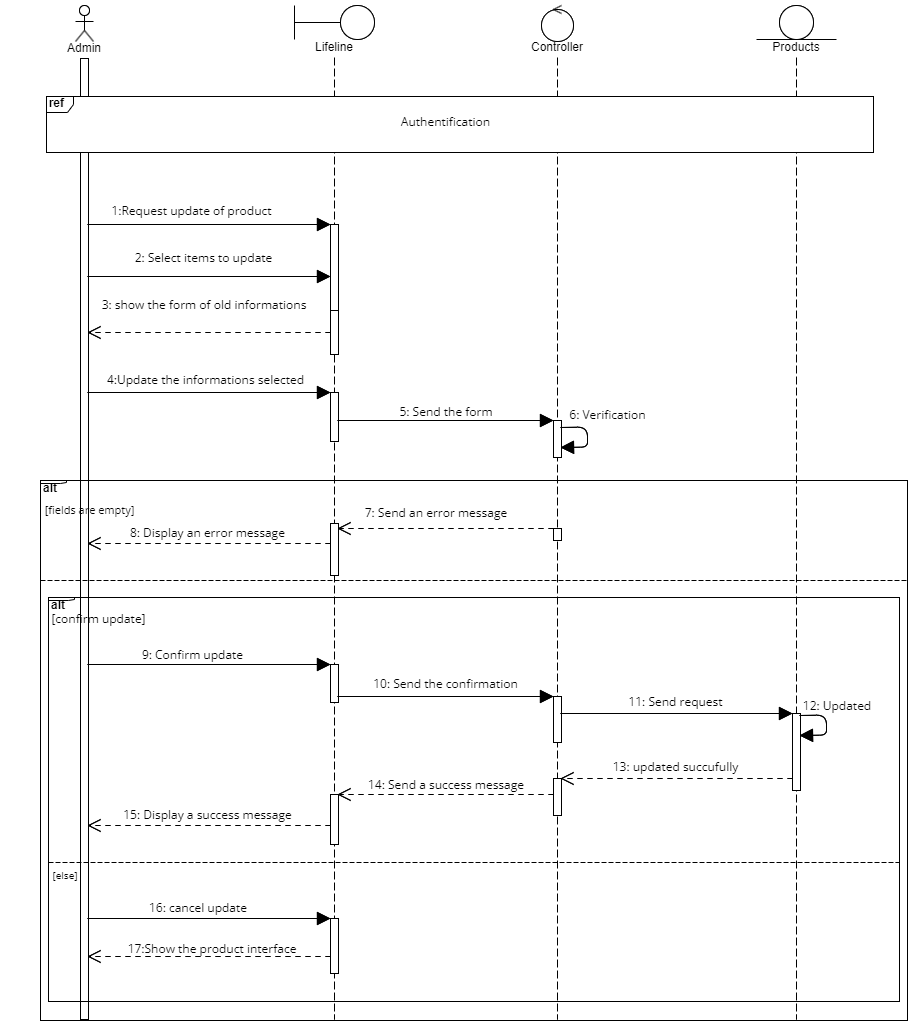


Figure 51:Modify product sequence diagram

The figure 44 represents the sequence diagram for the use case « Manage Product : modify » of the admin.

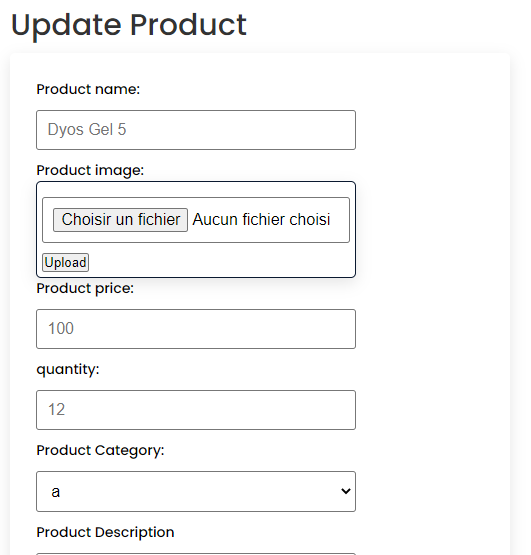


Figure 52:update product

**3.Refinement of use case «Manage Products: Delete product»:**

Table 12:Refinement of use case manage products.

|  |  |
| --- | --- |
| Use case | Authentication |
| Actor | Admin |
| Precondition | Admin must be logged in |
| Condition | Admin is logged in |
| Scenario | [Begin]  1.The admin access to the My Products tab.  2.the system displays Product interface.  3.the Admin chose the chose Delete product.  4.the Admin click "confirm".  [End] |
| Alternative | The system displays an error message if case existing product. |

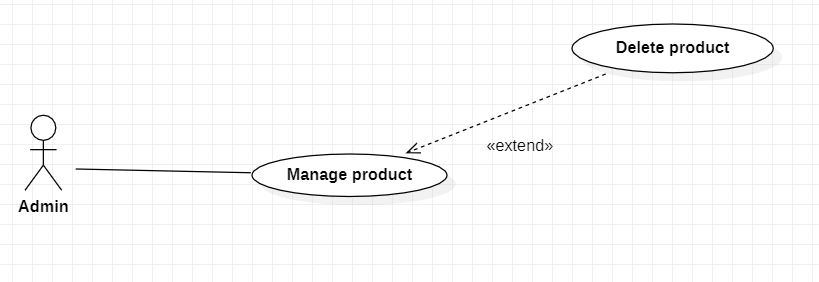


Figure 53:use case diagram of delete product

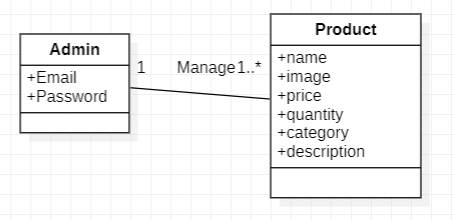


Figure 54:Global case of manage product

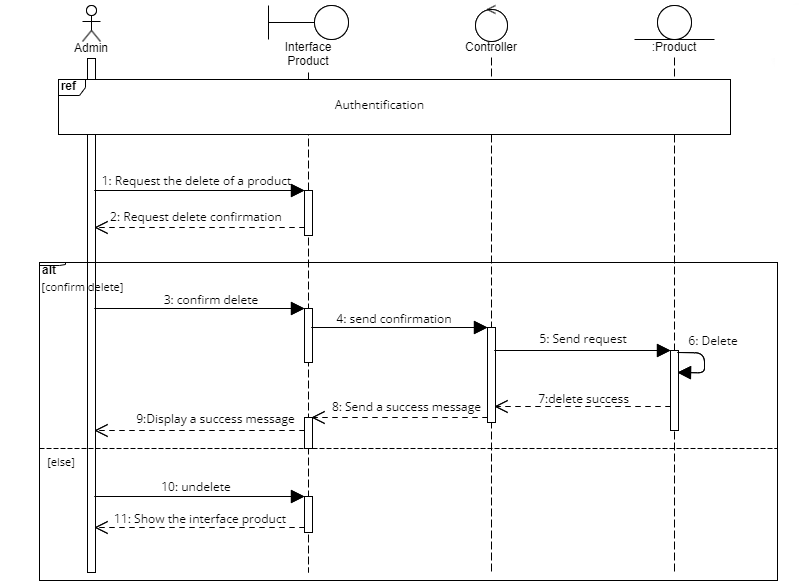


Figure 55:delete product sequence diagram

The figure 46 represents the sequence diagram for the use case « Manage Product Delete product » of the admin.

2.Realisation :

Figure 56 :Diagram sequence delete product