Computer Science Department

Faculty of Exact Sciences and Computer Science, University of Jijel

Application Development Report

Realized by: Hamza Chera, Mamoune Khaldi, Akram Rida, Ahmed Cherif Boumehraz, Mohcin Chettab, Zakaria Menad

Supervised by: Dr.El Hillali Kerkouche

December 21, 2024

Abstract

A report about the realization of an application for the management of a Medical Laboratory. This application is developed using the Java programming language and the JavaFX framework and follows a client-server architecture. The application allows the management of patients and their sample test results. It also allows the management of the laboratory's supply, finance, and staff.

Contents

1 Introduction

1.1 The Purpose Of This Work

The purpose of this work is to develop an application for License final year graduation project. The purpose of the application is the management of Medical analysis laboratories.

The application addresses the main functionalities needed for a smooth workflow in the laboratory. It aims to reduce the time and effort of the staff by providing easy to use management tools and straight forward user interfaces.

According to a thorough study of needs that we conducted looking at the essential functionalities used in the current medical laboratories' software, we came up with a list for what our application should aim to achieve.

- **Staff management**, allowing the easy management of roles and privileges.
- **Patients and sample management**, providing a well-designed system to store, manage and manipulate patients' data and their relevant documents and results.
- **Supply and finance management**, keeping the needed supply and financial flow in check and easily handled.

We judged that the above-mentioned functionalities are the most fundamental in the management of any Medical Analysis Laboratory, and we channeled our focus to create dependable solutions for each system.

1.2 Existing Solutions And Their Shortcomings

Upon looking at existing software solutions, their structure, and their shortcomings. We outlined some basic takeaways.

First, the dominant structure in most laboratories software is the client-server architecture. Which is reasonable considering the scale and nature of the project. The need for reliability, security and central management and control makes the client-server architecture perfect for this kind of medium-sized application and environment.

Secondly, we took notice that most software application that are under use simply look terrible. The user interface and user experience are neglected almost completely, so most of them seem unusable and likely require some training for a fresh eye.

The other thing that caught our attention during our inspection of the popular software options in the market is that most of them are overcomplicated. There are many unnecessary functionalities that are stuffed into the software with no real use in the day-to-day workflow of the lab.

In conclusion, we found a real lack of attention to the user experience and user interface in the existing software solutions. In addition to an overestimation of the functionalities needed versus the ones that are used in reality.

As a result, we decided to base our work on the fundamental client-server architecture for its convenience in aspects like security and the ability to share resources. While also filling the current gap in mainstream software by focusing on simplicity & efficacy in design choices, and improve the usability and user experience for the future user.

2 Requirements Analysis

2.1 Introduction

The requirements analysis is a crucial step in the development of any software application. It is the process of identifying the needs and constraints of the system to be developed. It is the first step in the software development life cycle and it is the foundation for the design and implementation of the system.

The process we embraced in identifying the needs of everyone that will be depending on the application (e.g., laboratory staff, administrators, patients) was careful and relied on frequent visits to different local laboratories and asking questions about the specific workflow of the laboratory. As well as inspecting the software used there.

We used the collected information to construct a contextual and realistic scenario of the daily happenings in the laboratory based on the interviews and observations we made.

The scenario served as the basis for the identification of the different stakeholders and their needs.

2.2 Context & Scenario

1. A patient arrives at the laboratory. He is identified by (ID, Name, Surname, Birthday, Gender, Phone, Email, Ville). He gives his information to the receptionist. Three cases are deduced here:
   - First visit of a new patient: the receptionist inserts the patient's details, prints the invoice and a label containing the patient's first and last name, and hands them to the patient.
   - An old patient requests new blood work.[step 2]
   - An old patient is looking for his blood work results.[step 8]
2. The customer remains in the waiting room until their turn comes.
3. When it is their turn, the customer enters the sampling room and hands the label to the nurse.
4. The nurse draws the blood, places it in a tube dedicated to the type of analysis to be performed then attaches the label to the tube to obtain a sample.
5. The nurse takes the sample to the analysis room. The biologist then adds the necessary reagents to the sample, and places it in a special device called an "Automata".
6. The Automata displays the results of the analysis. The results obtained are inserted and printed by the biologist.
7. The doctor validates the results, the receptionist collects the report and places it in an envelope.
8. Finally, the patient takes the report and pays the invoice, and the receptionist validates the payment.

Note 1: The lab manager is able to add, modify and delete an employee defined by (ID, Name, Role, Username, Password). He is also able to consult the state of the stock and the financial situation of the laboratory.

2.3 Stakeholders & Specifications

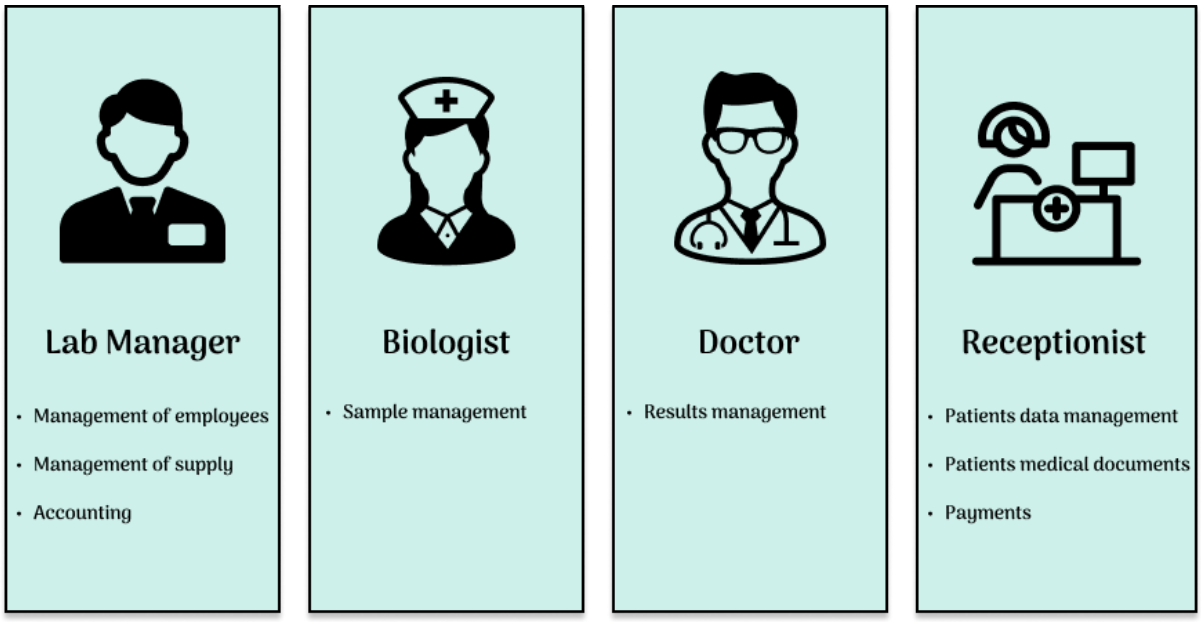

Figure 1: Stakeholders and their specifications

Laboratory Manager:

The laboratory manager is responsible for the overall management of the laboratory. He is responsible for the management of employees, supply, and accounting.

The functionalities needed by the laboratory manager are:

- **Management of employees:** a system for the management of the roles and privileges of each employee and the resources they can access. An employee is defined by (ID, Name, Role, Username, Password, Phone, Email).
- **Management of supply:** the ability to check, and control (add, modify and remove) the many equipments (tubes, reagents …) that are used on a daily basis.
- **Accounting:** a system to keep track of the different spendings and earnings of the lab.

Biologist:

The biologist is responsible for the process of sample analysis. He takes the collected samples and runs the required tests on them. He then inserts the obtained results into the system.

The functionalities needed by the biologist are:

- **Sample management:** the ability to view, add, modify the sample test results of patients on and from the database.

Doctor:

The doctor is responsible for viewing and checking the validity of the test results conducted by the biologist.

The functionalities needed by the doctor are:

- **Results validity management:** the ability to view the results of each patient's test and choose to validate those results for final delivery or not.

Receptionist:

The receptionist is responsible for the management of patients' data, medical history, and payments.

The functionalities needed by the receptionist are:

- **Patients data management:** a system for storing and manipulating patients data. It should provide the ability to add new patients and to modify or delete existing ones.
- **Patients medical documents:** the ability to input and keep track of patient's medical history and sample test results.
- **Patients payments:** keeps track of the payment situation of each patient and provides essential information about it.

2.4 Conclusion

In this section we were able to construct a descriptive scenario of the daily workflow in a medical laboratory. We used this scenario to identify the different stakeholders and their needs. We then specified the functionalities needed by each stakeholder. In the next section, we will use this information to inform our design choices and to develop the application.

3 Application Design

3.1 Introduction

In engineering, the design phase is aimed at formalizing the preliminary stages of developing a system to make this development more faithful to the client's needs.

Design is a crucial step in the development of any software application. It is the process of describing the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It is the bridge between requirements analysis and implementation.

Its goal is to describe unambiguously, often using a modeling language, the future operation of the system to facilitate its implementation.

There are two main approaches to designing information systems: the functional approach and the object-oriented approach. The functional approach perceives the system as an entity performing a global function that can be broken down into sub-functions. The object-oriented approach views the system as a set of objects interacting with each other.

In our work, we have opted for the object-oriented approach, which ensures software evaluation and object realization, which is not guaranteed with simplified modeling and development methods.

3.2 Unified Modeling Language

Unified Modeling Language (UML) is a standardized modeling language used in software engineering for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. It is a graphical language that provides a common vocabulary and set of diagrams to describe software systems.

3.3 Why UML?

- **Visual Representation:** UML provides a visual representation of the system which is easy to understand and interpret.
- **Standardization:** UML is a standardized language for specifying, visualizing, constructing, and documenting the artifacts of software systems. This standardization helps in maintaining consistency across different projects.
- **Communication:** UML diagrams facilitate effective communication among team members and stakeholders. They provide a common language that all parties involved can understand, reducing the chances of miscommunication.
- **Design Validation:** UML diagrams can be used to validate the design of the system. They can help identify potential issues or flaws in the design before the implementation phase.
- **Documentation:** UML diagrams serve as a form of documentation for the system. They can be used as a reference for future development or maintenance activities.
- **Modularity:** UML promotes modularity and separation of concerns by allowing the design to be broken down into different diagrams each focusing on a specific aspect of the system.
- **Platform Independent:** UML is platform independent. It can be used to design systems irrespective of the programming languages and platforms used for implementation.

3.4 Diagrams

Use Case Diagram:

Definition: Use case diagrams are a type of behavior diagram in UML that represent the functional requirements of a system from the perspective of its users. They show the interactions between the system and its actors (users or external systems) by depicting the various use cases (functions or services) that the system provides. Use case diagrams help in identifying, organizing, and prioritizing the functionalities of a system and are often used during the requirements' analysis phase of software development.

Purpose:

- Define the limits of the system.
- Define the system environment: the users and elements that interact with the system.
- Define the fundamental usages of the system.

Diagram Elements:

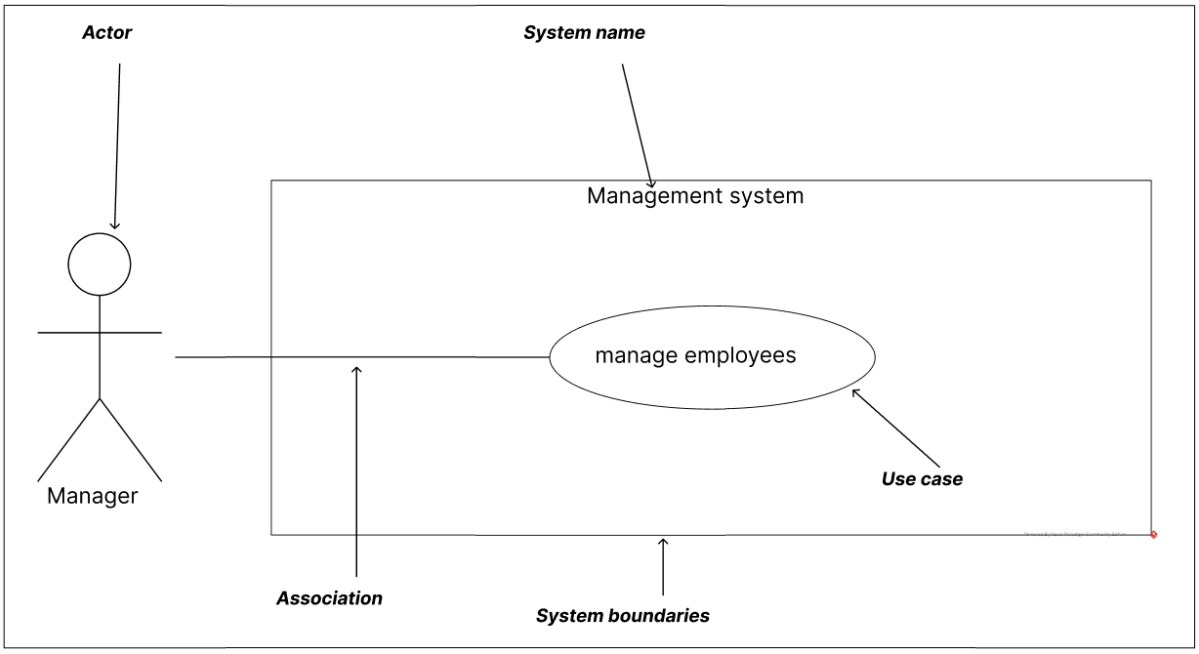

Figure 2: Use Case Diagram Elements

- **Actor:** An actor represents a role played by a user or any other system that interacts with the system. Actors are drawn as stick figures and are connected to use cases by lines.

Figure 3: Representation of ”Actor” in a Use Case Diagram

- **Use Case:** A use case represents a specific functionality or service provided by the system. It describes the interactions between the system and its actors to achieve a specific goal. Use cases are drawn as ovals and are connected to actors by lines.

Figure 4: Representation of ”Use Case” in a Use Case Diagram

- **Associations:** There are three types:
  - Between actor and use case: it is simple, connecting an actor and its functionalities.
  - Between two use cases: connecting two use cases have a correspondence. Represented in tree forms depending on the case; (specialization/generalization, inclusion, extension).

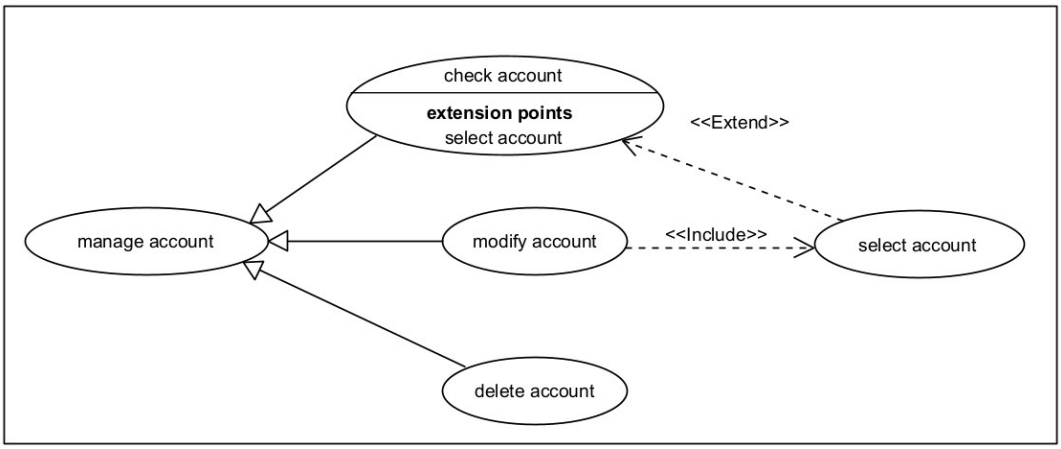

Figure 5: Different types of associations in a Use Case Diagram

  - Between actors: connects two actors, having an inheritance relationship between them.

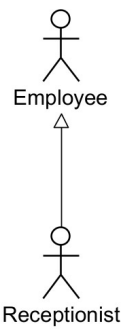

Figure 6: Association between two actors in a Use Case Diagram

In our project, we opted for using a structured Use Case Diagram for each actor of the system.

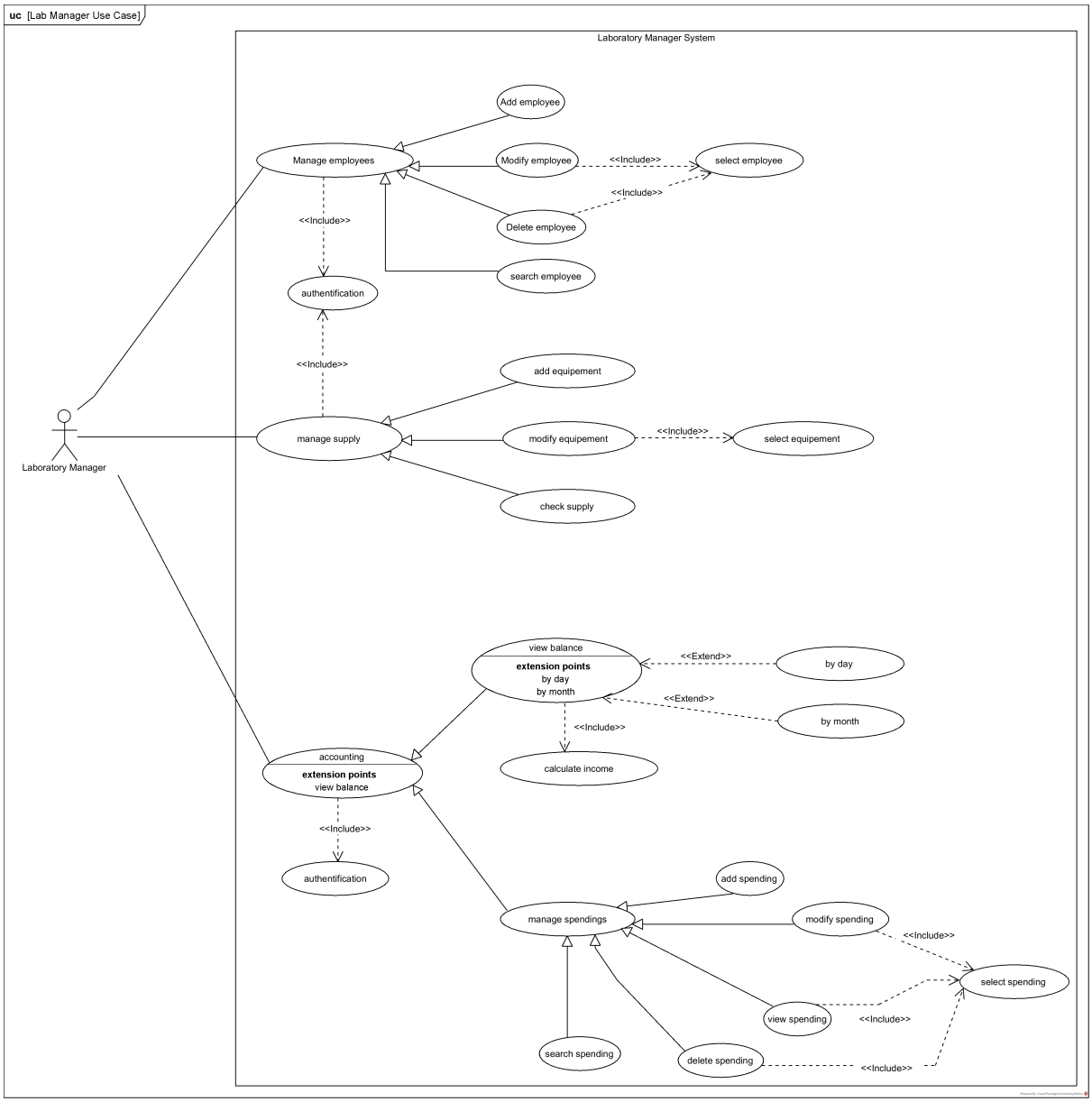

Figure 7: Use Case Diagram for the Laboratory Manager

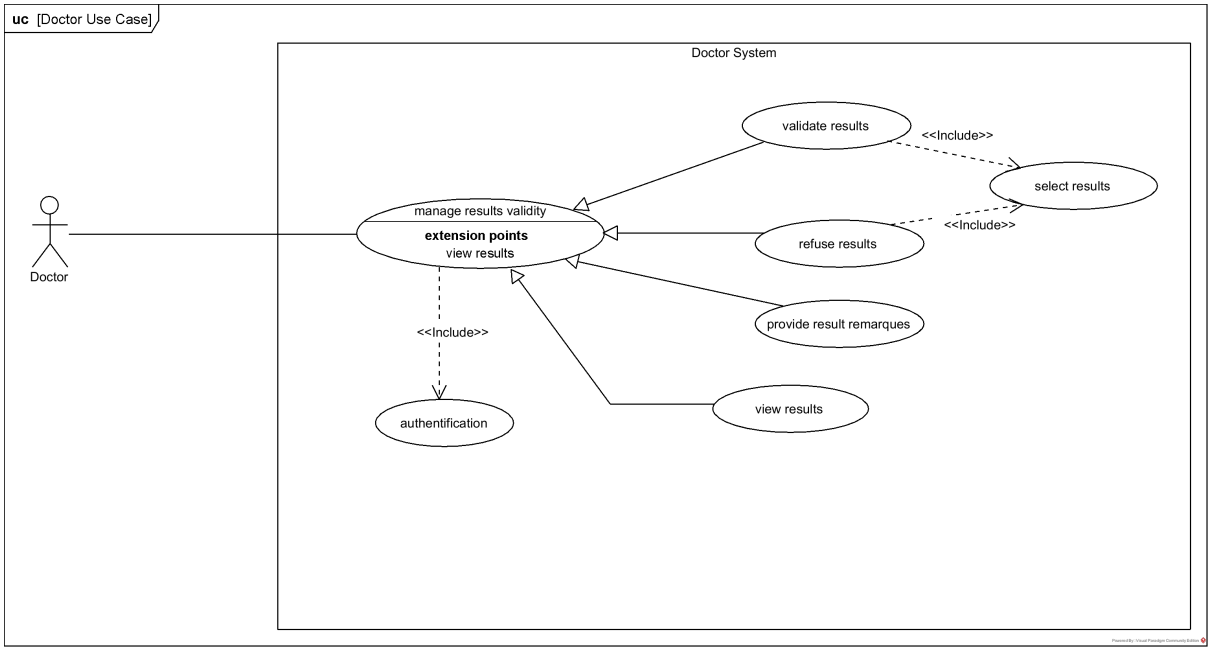

Figure 8: Use Case Diagram for the Doctor

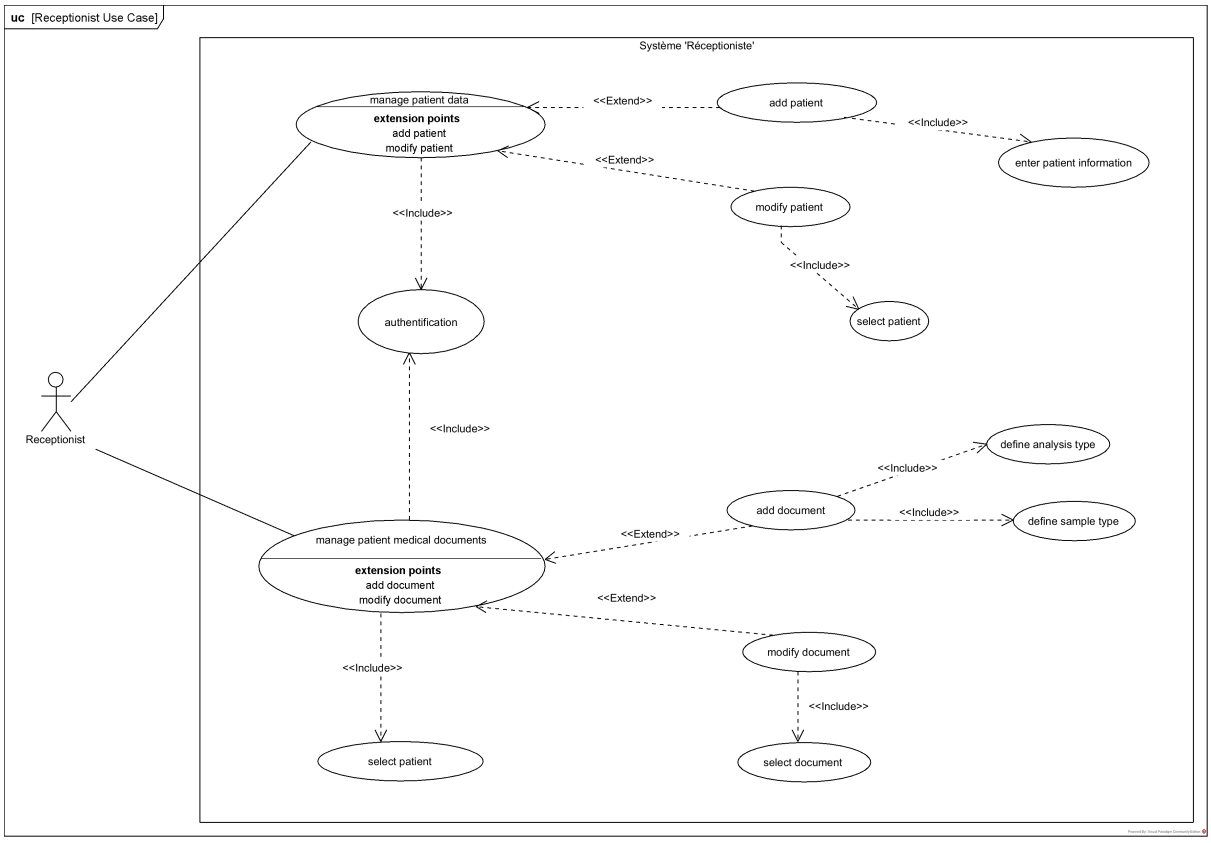

Figure 9: Use Case Diagram for the Receptionist

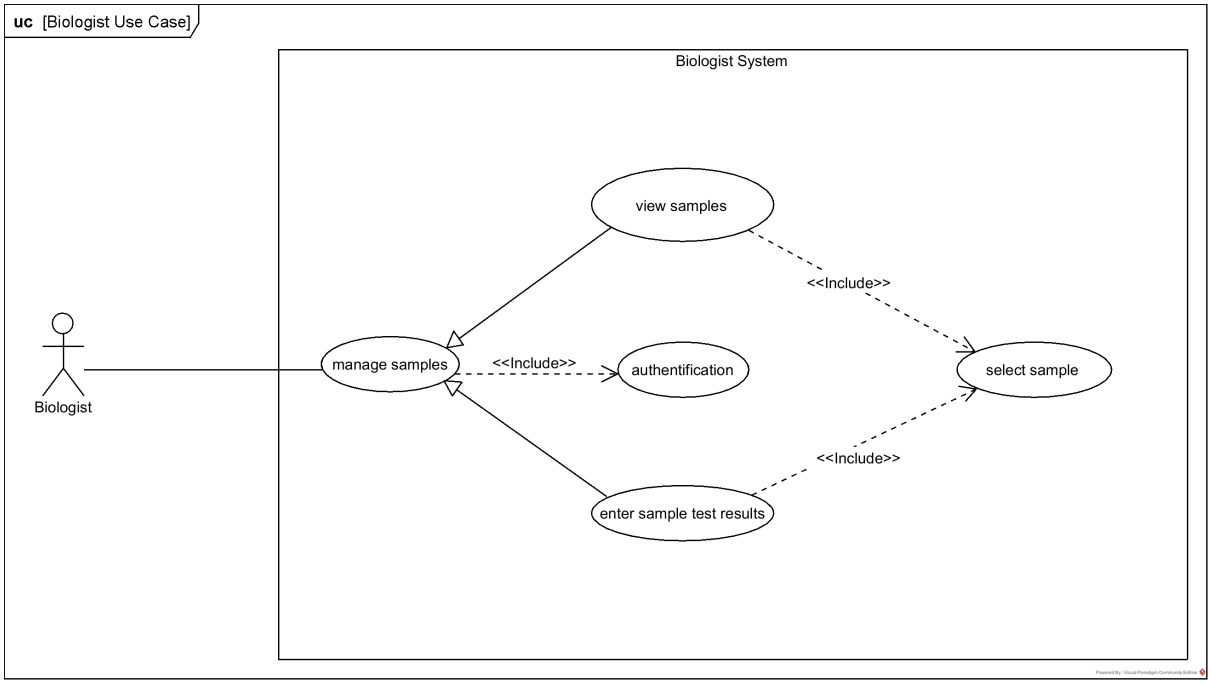

Figure 10: Use Case Diagram for the Biologist

Descriptive Cards of Use Cases

The description of a Use Case is a narrative document that describes the sequence of interactions in which the actor uses the system to arrive at a result that satisfies the initial purpose of the Use Case.

Here are our descriptive cards for the main use cases: Descriptive Card for "Authentication":

Use Case Name: Authentication
Type: Secondary
Actor: Laboratory manager, Biologist, Doctor, Receptionist
Purpose: Allows the actors to access their respective application interface, the user needs to enter their user name and password, then the system checks if the credentials are correct or not for the purpose of giving or denying their access request.
Precondition: The user already has an account.
Scenario: **Start Action:** The user starts the application and is welcomed by the login interface. **Ongoing Action:** • The user enters their username and password and clicks on the login button. • The system checks the credentials. • The system grants or denies access. **Final Action:** The system redirects the user to their interface.
Alternative: • The user enters incorrect credentials. • The login fields are empty.
Exception: • the user is not registered. • cancellation of the login.

Table 1: Descriptive Card for "Authentication"

Descriptive Card for ”Add employee”:

Use Case Name: Add employee
Type: Primary
Actor: Laboratory manager
Purpose: Allows the laboratory manager to add a new employee (user) to the system along with their information.
Precondition: • The laboratory manager is logged in.
Scenario: **Start Action:** The laboratory manager clicks on the ”Add employee” button. **Ongoing Action:** • The laboratory manager fills in the employee’s information (ID, Name, Role, Username, Password, Phone, Email). • the laboratory manager clicks on the ”Add” button. • The system adds the employee to the database. **Final Action:** The system confirms the operation by displaying the new list of existing employees
Alternative: • the employee already exists. • the fields are empty.
Exception: • the operation is canceled.

Table 2: Descriptive Card for ”Add employee”

Descriptive Card for ”Search employee”:

Use Case Name: Search employee
Type: Primary
Actor: Laboratory manager
Purpose: The purpose of the employee search in a laboratory analysis system is to provide quick, accurate, and secure access to employee information and to verify their existence in the system.
Precondition: • The laboratory manager is logged in. • The employee exists in the system.
Scenario: **Start Action:** The laboratory manager clicks on the search bar. **Ongoing Action:** • The laboratory manager enters the employee’s name or ID. • The system filters the list of employees. **Final Action:** the system displays the employee information.
Alternative: • incorrect employee name or ID. • empty search bar.
Exception: • the operation is canceled.

Table 3: Descriptive Card for ”Search employee”

Descriptive Card for ”Modify equipment”:

Use Case Name: Modify equipment
Type: Primary
Actor: Laboratory Manager
Purpose: Allows the laboratory manager to change the information of an existing equipment in the supply management interface. It allows the modification of the equipment’s name, quantity and type.
Precondition: • the laboratory manager is logged in. • the equipment already exists.
Scenario: **Start Action:** the laboratory manager select the equipment and clicks the ”modify” button. **Ongoing Action:** • the laboratory manager enter the new information in the appropriate fields • the laboratory manager clicks the ”save” button. • the system updates the equipment’s existing information. **Final Action:** the system displays the new product’s information
Alternative: • the entered information type is not compatible.
Exception: • the operation is canceled.

Table 4: Descriptive Card for ”Modify equipment”

Descriptive Card for ”View Balance”:

Use Case Name: View Balance
Type: Primary
Actor: Laboratory Manager
Purpose: Allows the laboratory manager to view the flow of income and spendings of the laboratory, as well as view the calculated net income.
Precondition: • the laboratory manager is logged in.
Scenario: **Start Action:** Clicks on the ”Finance” option in the side bar **Ongoing Action:** • the system displays the spendings and income transactions **Final Action:** The system calculate the net income and displays it to the user.
Alternative: • there are no transactions yet.
Exception: //

Table 5: Descriptive Card for ”View Balance”

Descriptive Card for ”Add patient medical document”:

Use Case Name: Add document
Type: Primary
Actor: Receptionist
Purpose: Allows the creation of a request for a medical test document associated with a patient which describes their analysis and their price; the receptionist must choose a patient then add their required analysis, and finally save the document to the database.
Precondition: • the receptionist is logged in. • Patient already exists in the database.
Scenario: **Start Action:** The Receptionist clicks on the ”New” button. **Ongoing Action:** • The system shows a window which contains the list of patients. • The receptionist chooses a patient. • the system shows the list of available analysis. • The receptionist chooses the analysis. • The receptionist clicks on the ”Save” button. • the system calculates the price. **Final Action:** the system saves the test request and relevant information in the database.
Alternative: • the patient or the analysis does not exist.
Exception: • the operation is canceled.

Table 6: Descriptive Card for ”Add document”

Descriptive Card for ”Validate Results”:

Use Case Name: Validate Results
Type: Primary
Actor: Doctor
Purpose: Allows the doctor to be able to validate the analysis results that are submitted by the biologist. It ensures the credibility of the results.
Precondition: • the doctor is logged in. • The results have been submitted by the biologist. • the doctor has the results code.
Scenario: **Start Action:** The doctor select the search bar. **Ongoing Action:** • The doctor enters the results code. • The doctor checks the validity of the results. • the doctor enters a comment. • The doctor clicks on the ”Validate” or ”Refuse” button. **Final Action:** The system saves the validation information.
Alternative: • the results code is incorrect.
Exception: • the operation is canceled.

Table 7: Descriptive Card for ”Validate Results”

Sequence Diagram:

Definition: A sequence diagram is a Unified Modeling Language (UML) diagram that represents the sequence of messages between objects during an interaction.

Purpose:

- Describes the implementation of use cases on the system described by the class diagram.
- An internal point of view into the system functions.
- Description at a momentary level (system state at a given moment).
- Representation of messages exchange.

Diagram Elements:

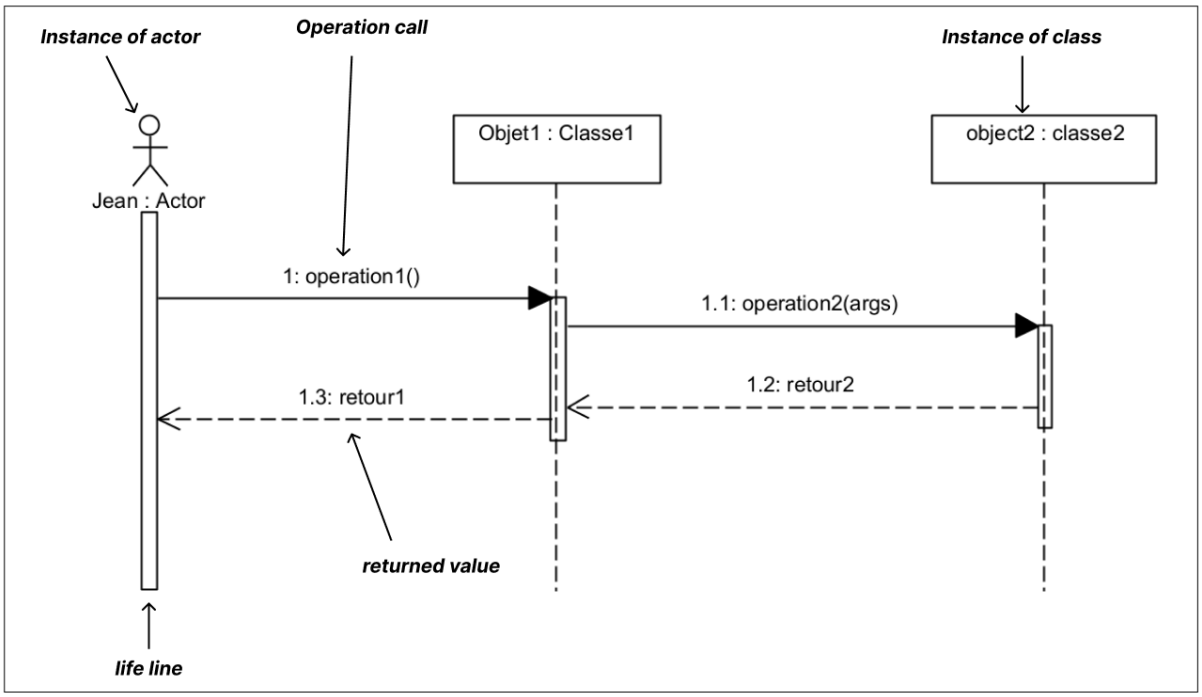

Figure 11: Sequence Diagram Elements

Different Operation Types:

- Alternative: Conditional message sending.
- Reference: Reference of an external diagram.
- Break: An exception raised.
- Loop: Repeat a sequence of messages.

Here are our Sequence Diagrams for the main functionalities of the system:

This Sequence Diagram represents the interactions between the user and the system to obtain an access authorization.

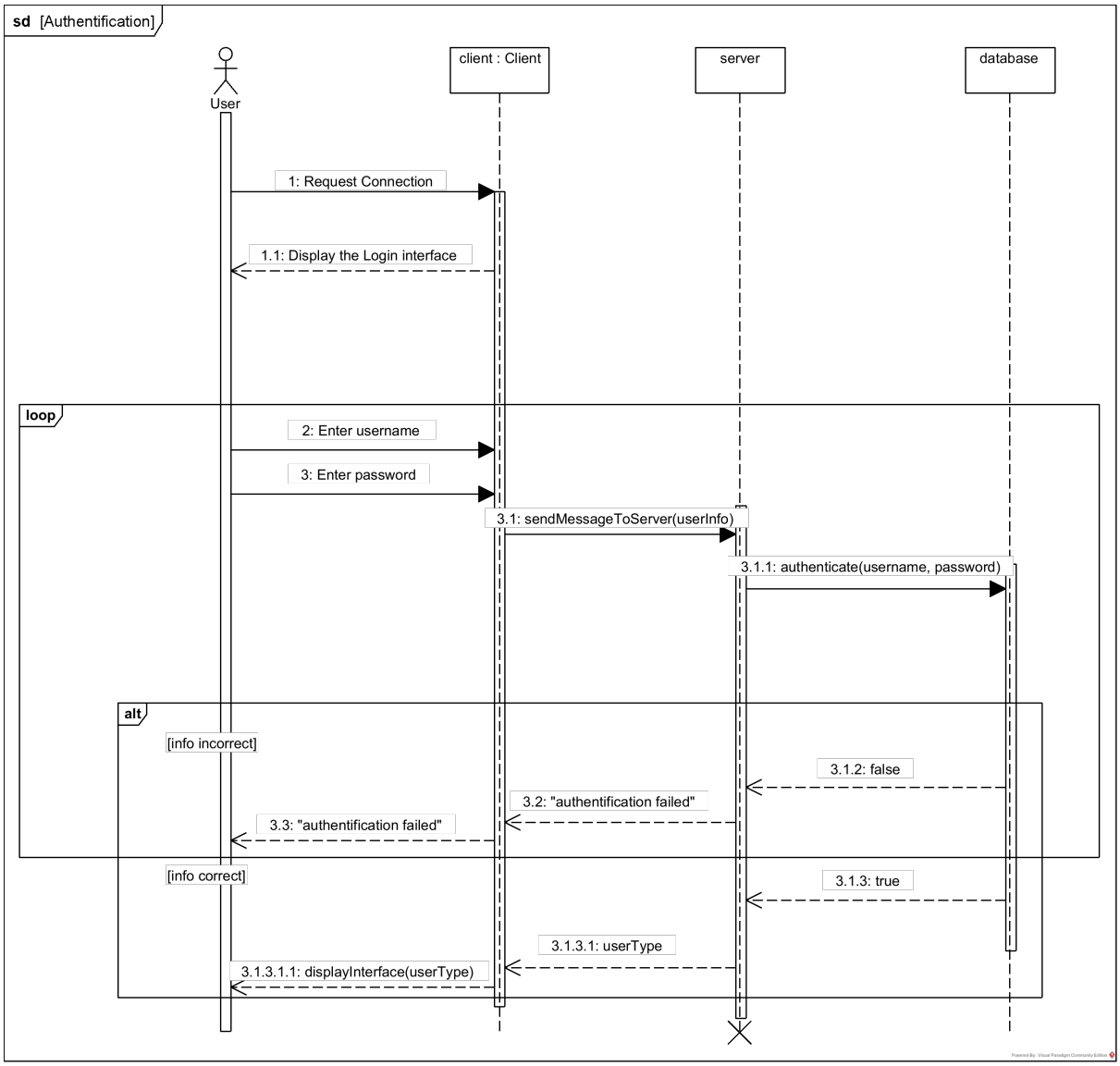

Figure 12: Sequence Diagram for the Authentication

This Sequence Diagram showcases the sequence of interactions between the user and the system in order to search for an existing user.

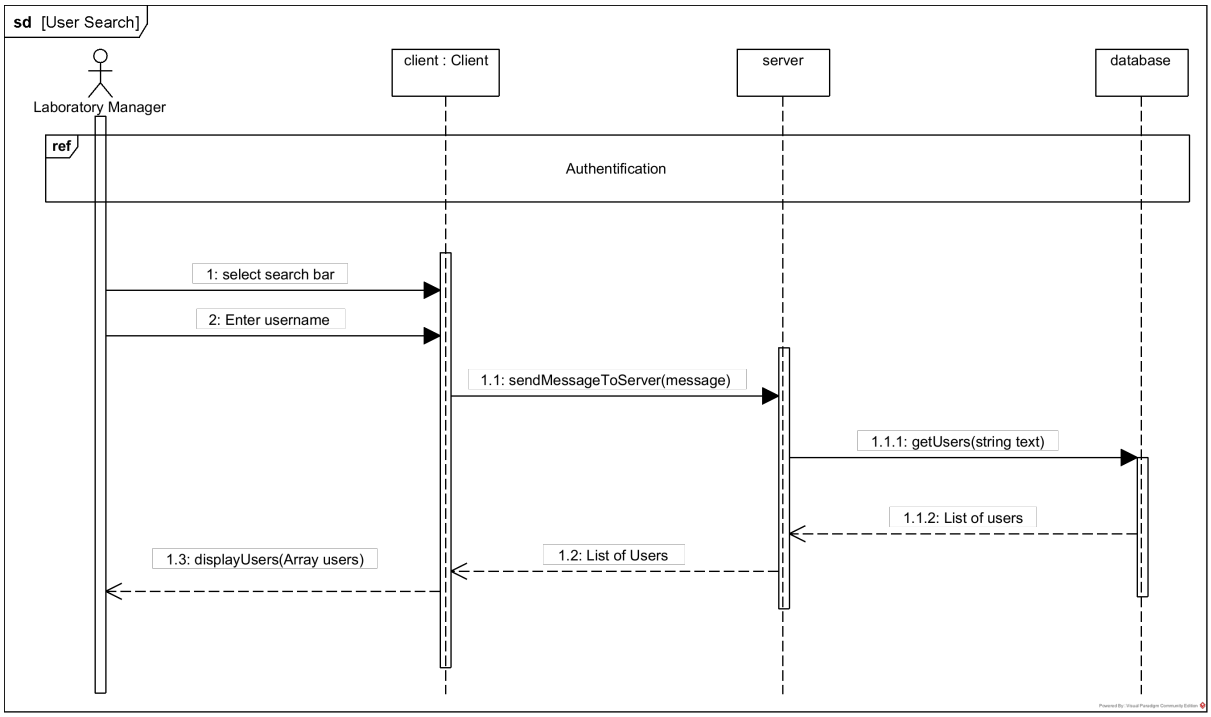

Figure 13: Sequence Diagram for searching a user

Note: A similar version of the sequence diagram in [Figure 13] is used for the search of a patient, a sample, an equipment, and a test result.

This Sequence Diagram explains the interactions between the user and the system components for adding a new user to the database.

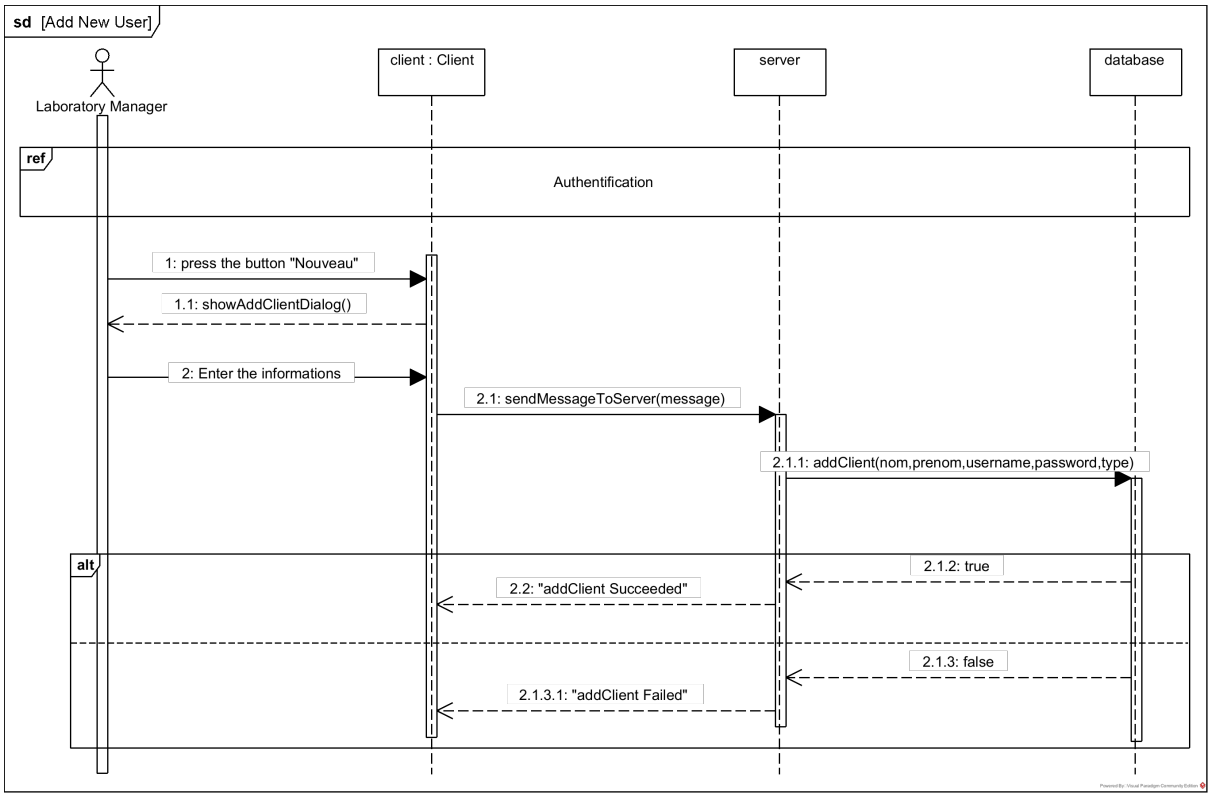

Figure 14: Sequence Diagram for adding a user

Note: A similar version of the sequence diagram in [Figure 14] is used for adding a patient, a sample request, an equipment, a payment or purchase.

This Sequence Diagram represents the user-system interactions for modifying the information of an existing user.

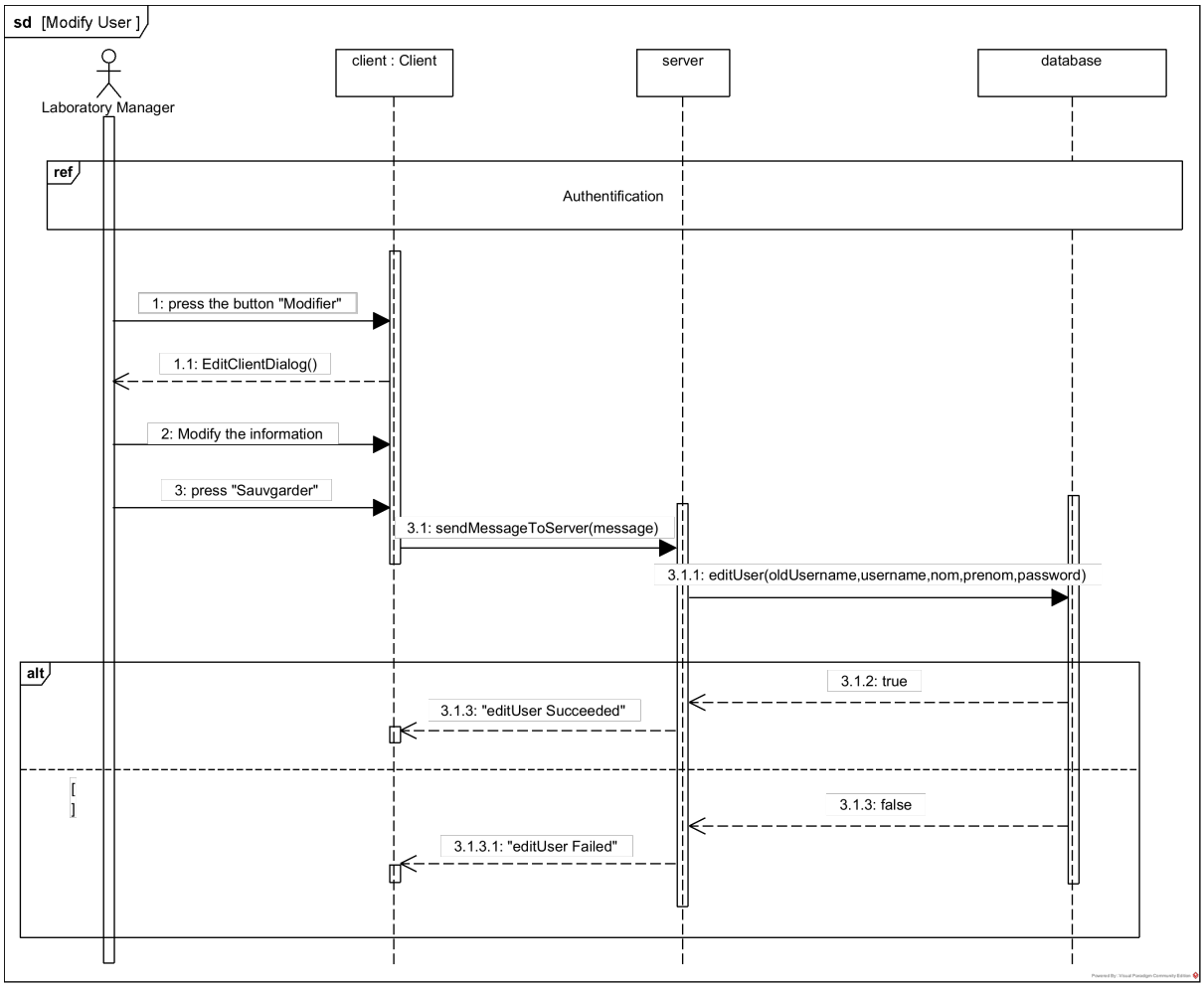

Figure 15: Sequence Diagram for modifying a user

Note: A similar version of the sequence diagram in [Figure 15] is used for modifying a patient, an equipment.

This Sequence Diagram details the various interactions between the user and the system for deleting an existing user.

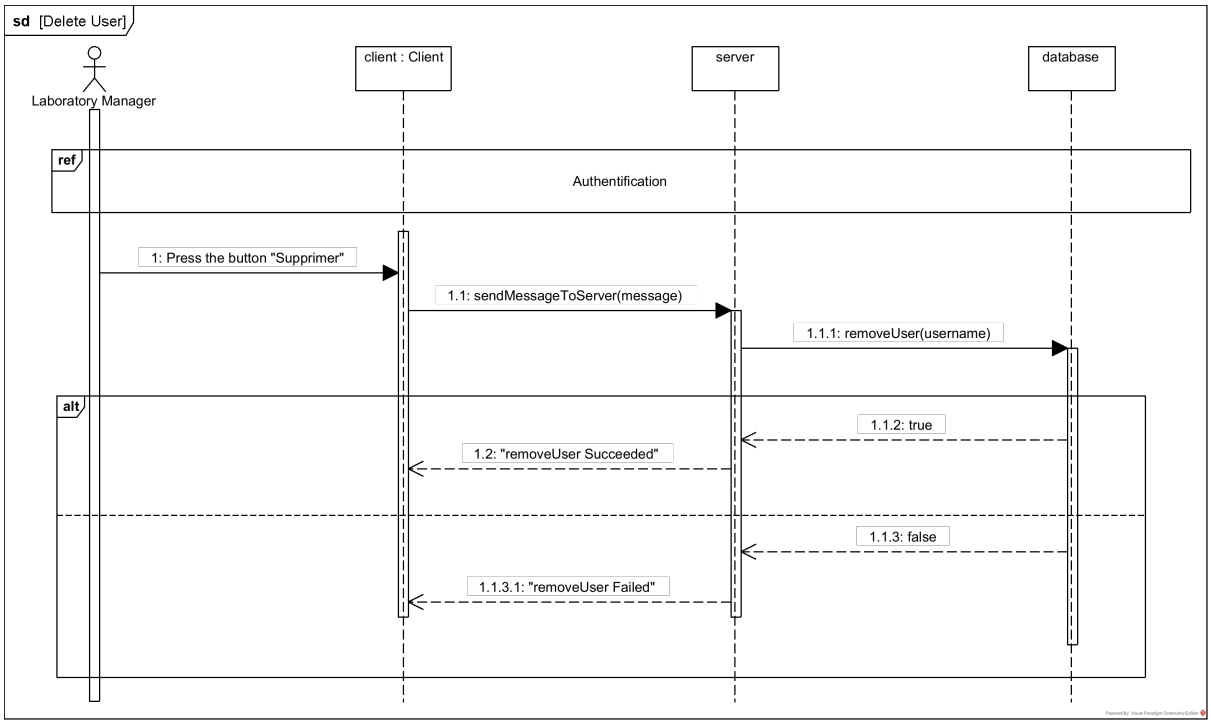

Figure 16: Sequence Diagram for deleting a user

Note: A similar version of the sequence diagram in [Figure 16] is used for deleting a patient, an equipment, a sample request.

This Sequence Diagram details the various interactions between the user and the system for adding analysis test results.

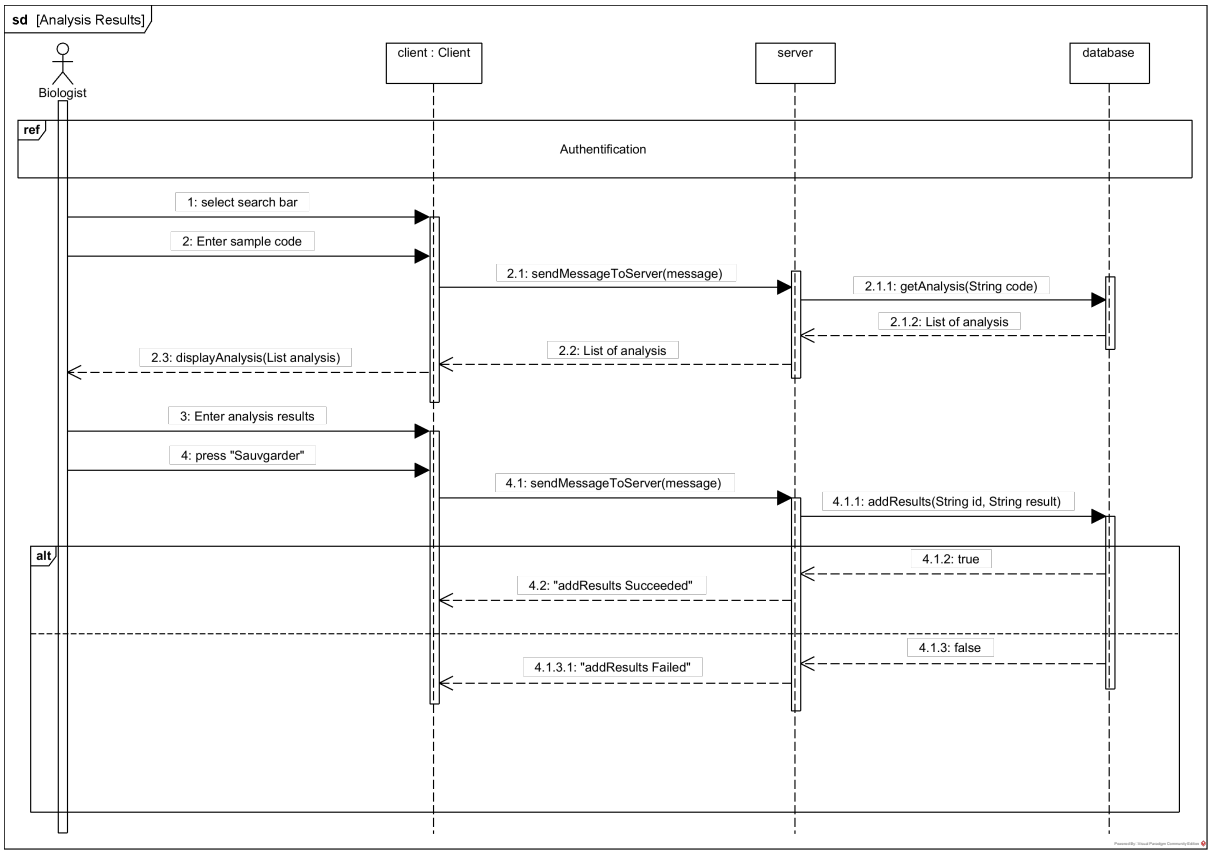

Figure 17: Sequence Diagram for adding analysis results

Note: A similar version of the sequence diagram in [Figure 17] is used for modifying an analysis results.

This Sequence Diagram details the various interactions between the user and the system for the validation of analysis results.

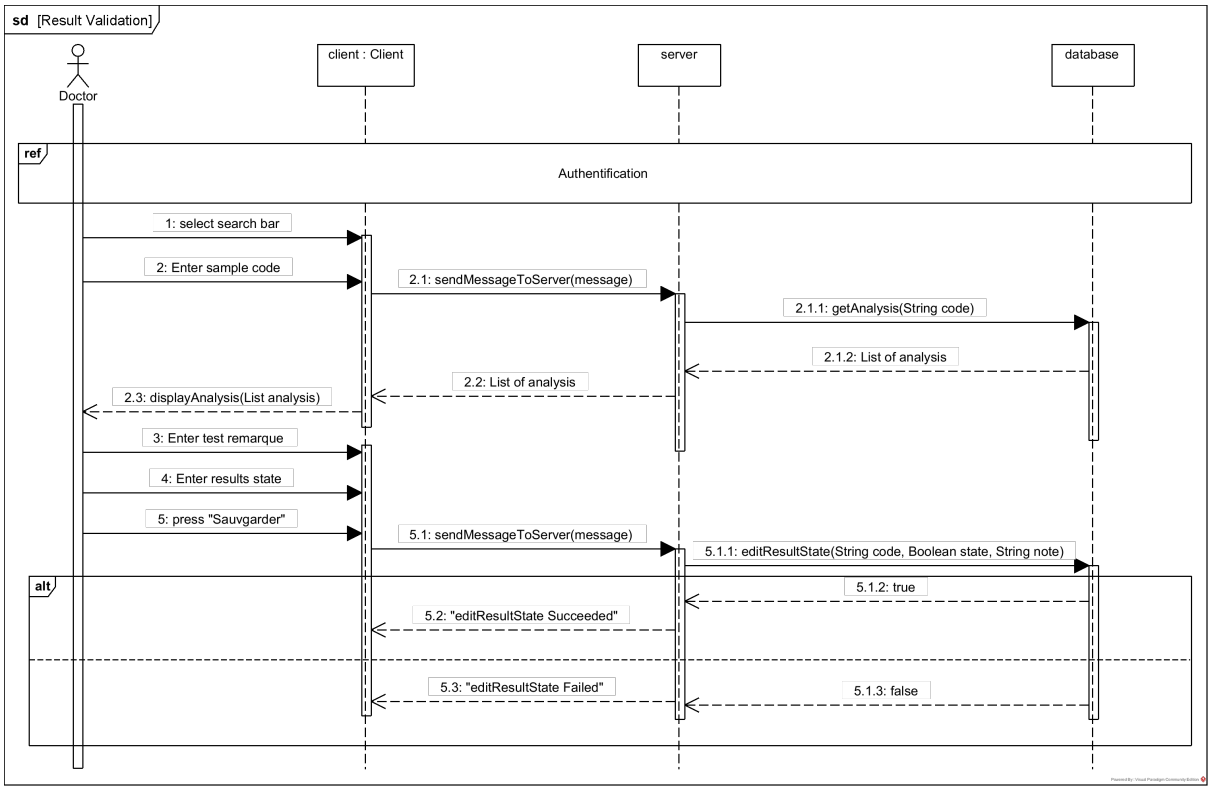

Figure 18: Sequence Diagram for analysis result validation

Class Diagram:

Definition: Class diagrams are a type of static structure diagram in UML that represent the structure of a system by showing the classes of the system, their attributes, methods, and relationships between them. They depict the static view of the system, focusing on the entities (classes) within the system and their relationships. Class diagrams are widely used in object-oriented analysis and design to model the structure of software systems.

Purpose:

- Represents the state of the software.
- Represents the internal structure.
- Diagram evolution with the execution of the software (the creation, deletion, and modification (relation, object)).

Diagram Elements:

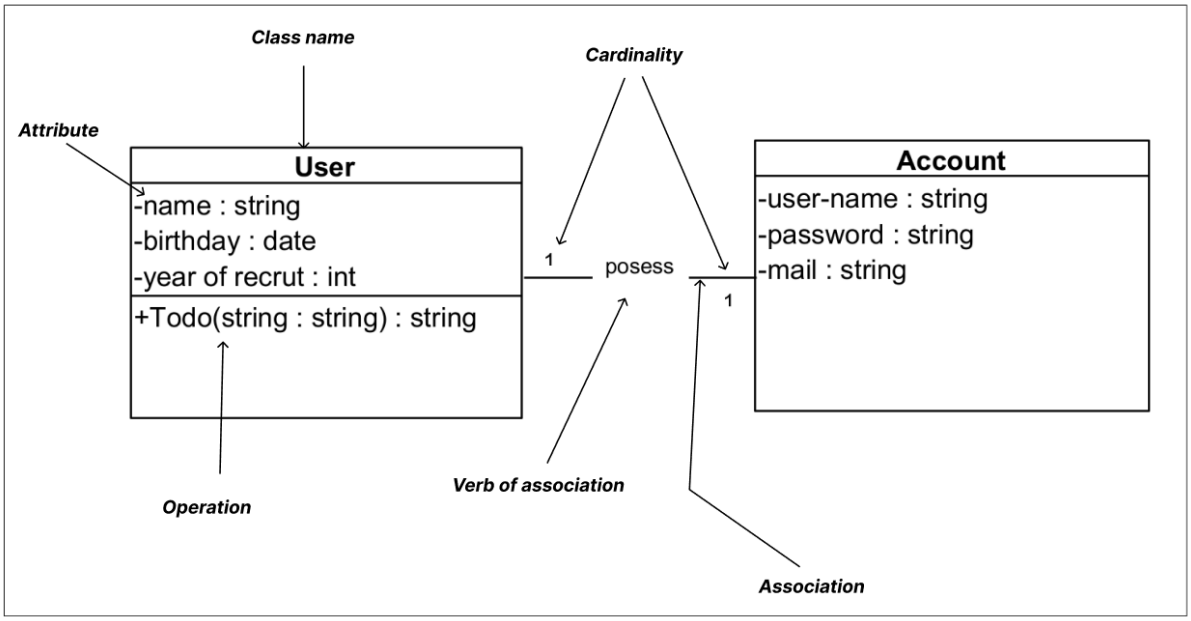

Figure 19: Class Diagram Elements

Here is our Class Diagram:

This diagram shows us the detailed class tables that are needed in our application.

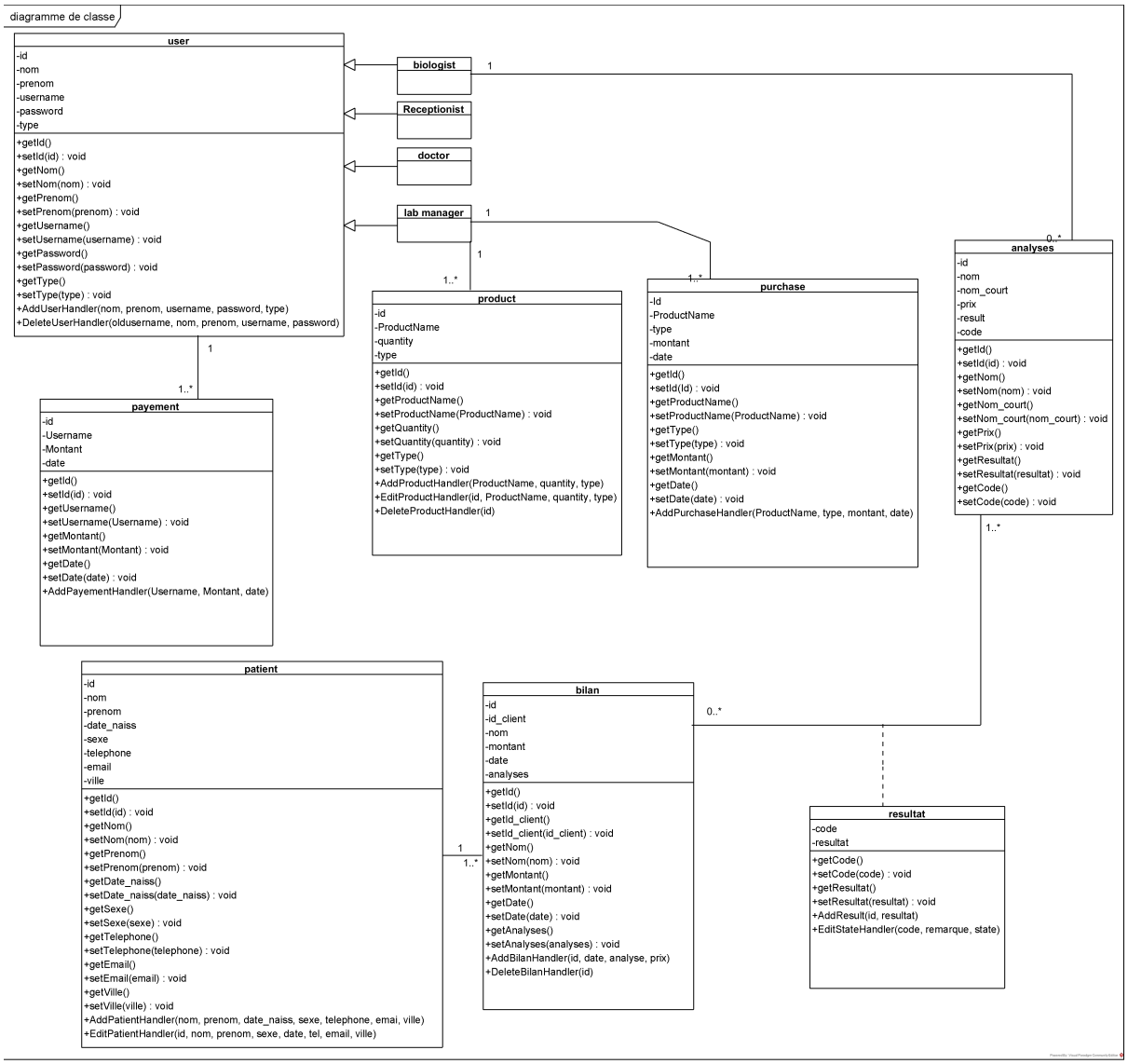

Figure 20: Detailed Class Diagram for our application

Entity Relationship Diagram:

Definition: An Entity-Relationship (ER) diagram is a type of data model that describes the logical structure of a database. It shows the relationships between entities and their attributes. ER diagrams are widely used in database design and are an essential tool for software developers, database administrators, and data analysts.

Purpose:

- Visualize the structure of a database.
- Identify the entities and their relationships.
- Understand the flow of data in a system.
- Design a database schema.

Diagram Elements:

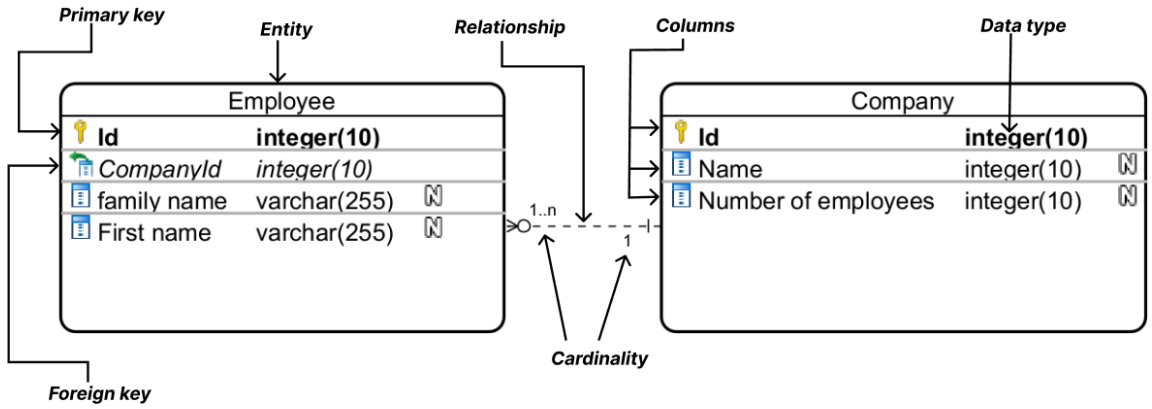

Figure 21: Entity Relationship Diagram Elements

Here is our Entity Relationship Diagram:

This Entity Relationship diagram represents the database tables that are used in our application.

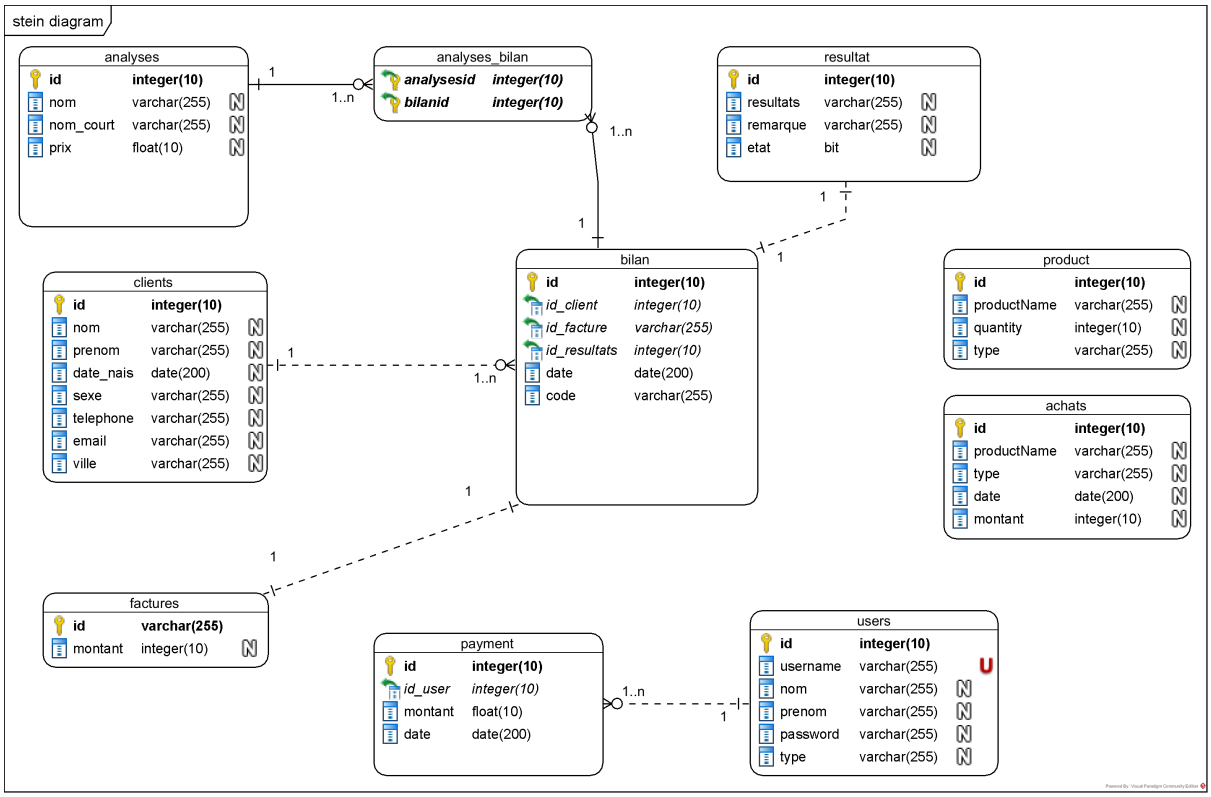

Figure 22: Entity Relationship Diagram for Application Database

3.5 ER Diagram to Relational Model Transformation

The transition from the DCL (Data Control Language) object model to the relational model is done with the aim of implementing the database as a relational DBMS (Database Management System). This transition is done by following a set of rules that allow the transformation of the object model into a relational model.

Rules of transformation:

- Each Entity is transformed into a table.
- Each column is kept as column.
- Each primary key is kept as primary key.
- Each relationship is transformed into a foreign key.

Relational Model:

- analyses (id, nom, nom_court, prix)
- clients (id, nom, prenom, date_nais, sexe, telephone, email, ville)
- bilan (id, id_client, id_facture, id_resultats, date, code)
- analyse_bilan (analysesid, bilanid)
- facture(id, montant)
- resultat (id, resultats, remarque, etat)
- product (id, productName, quantity, type)
- achats (id, productName, type, date, montant)
- users(id, username, nom, prenom, password, type)
- payment(id, id_user, montant, date)

3.6 Conclusion

In this section, we have used UML to design the application. We have created Use Case Diagrams, Sequence Diagrams, and Class Diagrams to represent the functionalities and structure of the system.

We have also transformed the Class Diagram into a relational model to implement the database. In the next section, we will use this design to implement the application.

4 Application Implementation

4.1 Introduction

After analyzing our needs and defining our project design methodology this section will be devoted to presenting the development tools and programming languages used for the implementation phase.

This phase is based primarily on the results of the design and analysis phase. Each use case, sequence and detailed sequence diagrams defined in the analysis and design phase are put to use in the implementation phase with the help of appropriate technologies.

These technologies will be presented in this section, showcasing the architecture of the platform and illustrate screenshots of the interfaces used.

4.2 Development environment

IntelliJ IDEA:

IntelliJ IDEA is an integrated development environment (IDE). It is a sophisticated and specialized IDE for Java and Kotlin development. It offers many tools and functionalities to maximize developer productivity.

We used IntelliJ IDEA as the main IDE for the development of our application.

Used version: IntelliJ IDEA 2024.1.1

4.3 Software

XAMPP:

The full form of XAMPP is “Cross-Platform (X), Apache (A), MySQL (M), PHP (P), and Perl (P).” This means that XAMPP is a tool that brings all these technologies together in one place.

Figure 23: What is XAMPP?

- **Apache:** Apache is a web server that displays your website on the internet. The Apache server that comes with XAMPP provides you with local development.
- **MySQL:** MySQL is an open-source relational database management system. You can use it to create a database and store data.
- **PHP:** PHP is a server-side scripting language that helps in creating dynamic web applications. XAMPP also provides you PHP environment.
- **Perl:** Perl is a programming language used for server-side scripting.

XAMPP is a popular open-source web server package for website development. This gives you cross-platform functionality that allows you to easily use it on any platform. XAMPP is used in web development to set up a complete web server. XAMPP Supports multiple scripting languages. In addition, XAMPP also has the functions of an FTP server, Accelerated Mode, network download, etc.

We used XAMPP to set up a local server for our application in order to simulate the client-server environment required for our application. We also used it to create and manage the database.

Used version: XAMPP 8.2.4

Figma:

Figma is a web-based collaborative interface design tool that empowers teams to create and iterate on user interfaces (UIs) and user experiences (UX) for web, desktop, and mobile applications.

We used Figma to design the user interfaces of our application.

Used version: Figma 116.18.6

Visual Paradigm:

Visual Paradigm offers a variety of diagramming tools that use the Unified Modeling Language (UML) to create visual representations of software systems. This includes things like flowcharts, use case diagrams, and class diagrams. These diagrams help developers understand, design, and document complex systems. It also goes beyond UML. While UML is a core strength, Visual Paradigm also supports other diagramming standards like mind maps and ER diagrams (entity-relationship diagrams) for broader project planning and data modeling.

We used Visual Paradigm for the application design phase to create UML diagrams.

Used version: Visual Paradigm CE 17.1

4.4 Programming languages

Java:

Java is a general-purpose, object-oriented programming language prized for its platform independence (write once, run anywhere) and focus on code reusability, security, and robustness. This makes it a powerful tool for developing a wide range of applications.

We used Java as the main programming language for the development of our application.

Used version: Java 17

JavaFX:

JavaFX is an open-source library for crafting desktop and rich internet applications (RIAs) with a focus on rich graphics, multimedia, and cross-platform compatibility.

We used JavaFX to create the graphical user interface (GUI) of our application.

Main Characteristics:

- **Rich Graphics:** JavaFX provides a set of graphics and media packages that enable developers to create rich, interactive applications.
- **Cross-Platform Compatibility:** JavaFX applications can run on any platform that supports Java, including Windows, macOS, and Linux.
- **Scene Builder:** JavaFX includes a visual layout tool called Scene Builder that allows developers to design user interfaces by dragging and dropping components onto a canvas.
- **Modern UI with FXML and CSS:** JavaFX utilizes FXML, a declarative language similar to HTML, for defining the application's user interface (UI). This simplifies UI development and separation of concerns. Additionally, CSS allows for extensive customization of the application's look and feel.

Used version: JavaFX 17.0.6

4.5 Database Management System

MySQL:

MySQL is an open-source relational database management system (RDBMS) that uses SQL (Structured Query Language) to manage and manipulate data. It is a popular choice for web applications and is widely used in conjunction with PHP to create dynamic websites.

We used MySQL as the database management system for our application with the help of XAMPP.

Used version: MySQL 15.1

5 Project Presentation

5.1 Application Interfaces

Login Interface

The main interface of the application is the login interface. It is where authentication takes place. The user must enter their username and password to access the application. The user will be logged in as a laboratory manager, a doctor, a receptionist, or a biologist depending on the credentials they enter.

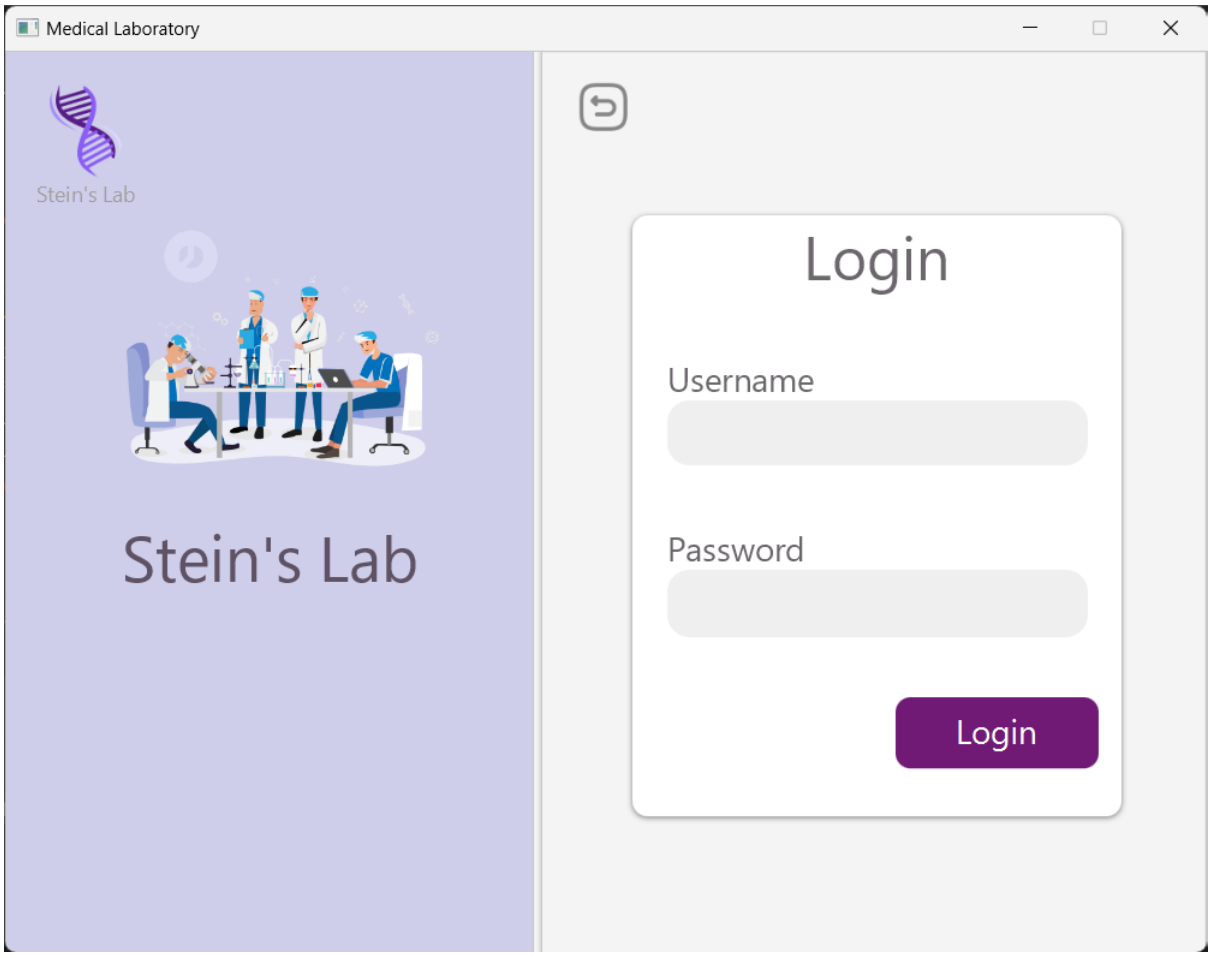

Figure 24: Login Interface

Management of Employees Interface

The management of employees interface allows the laboratory manager to manage the roles and privileges of each employee. The laboratory manager can add, modify, or delete an employee. The laboratory manager can also view the list of employees and their roles.

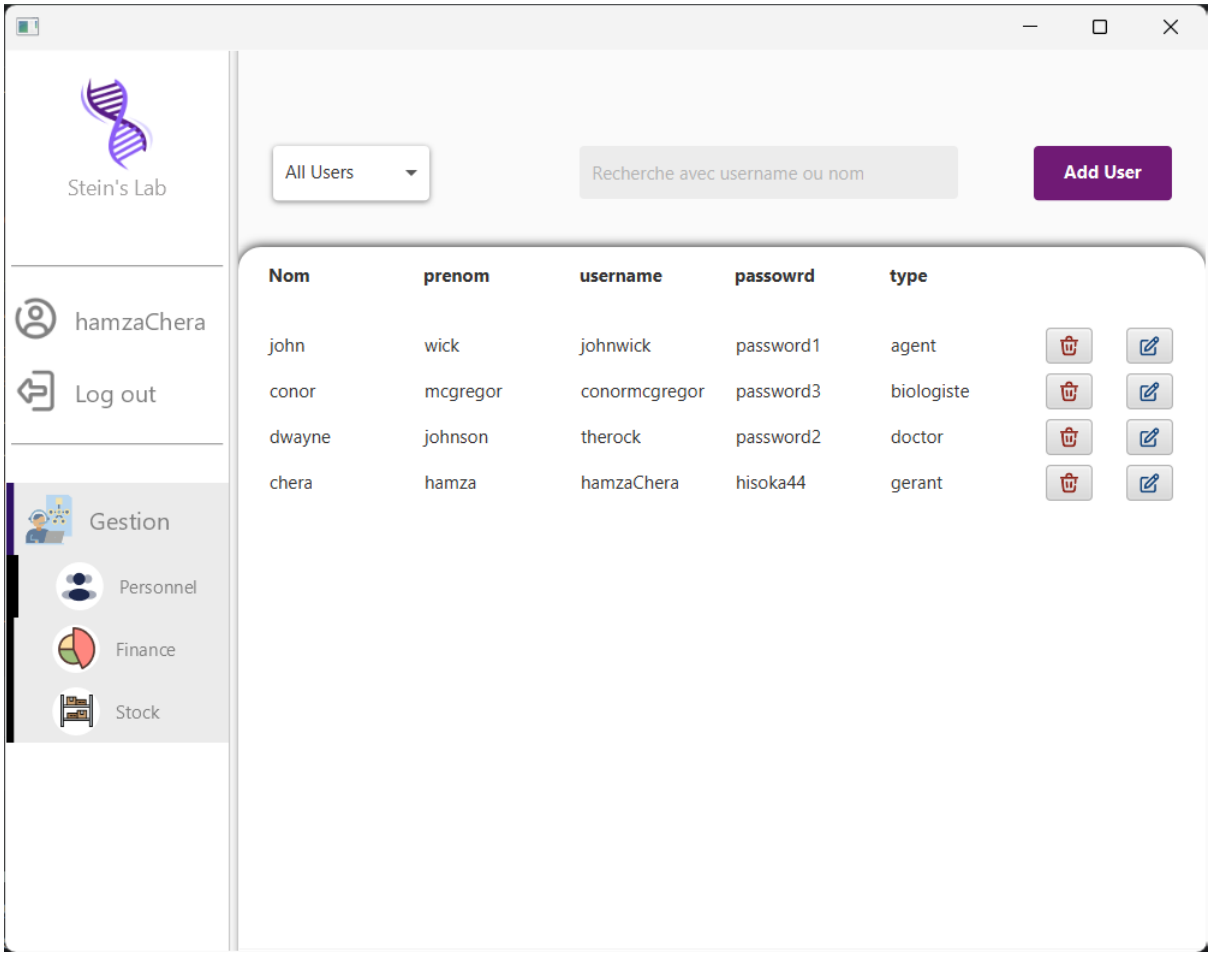

Figure 25: Management of Employees Interface

Management of Supply Interface

The management of supply interface allows the laboratory manager to manage the supply of the laboratory. The laboratory manager can add, modify, or delete an equipment. The laboratory manager can also view the list of equipment and their quantity.

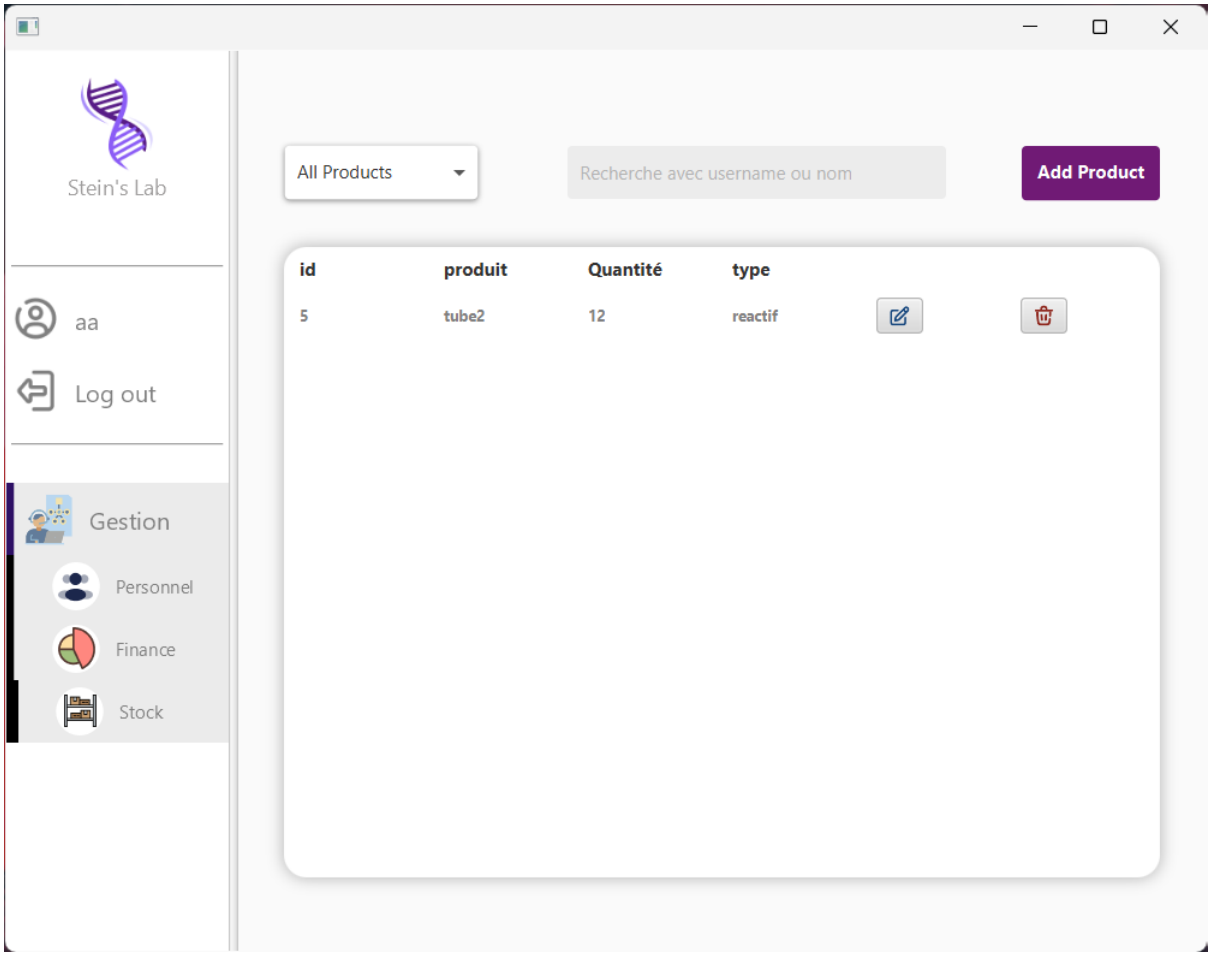

Figure 26: Management of Supply Interface

Accounting Interface

The accounting interface allows the laboratory manager to keep track of the financial situation of the laboratory. The laboratory manager can view the list of payments and purchases made by the laboratory as well as the income.

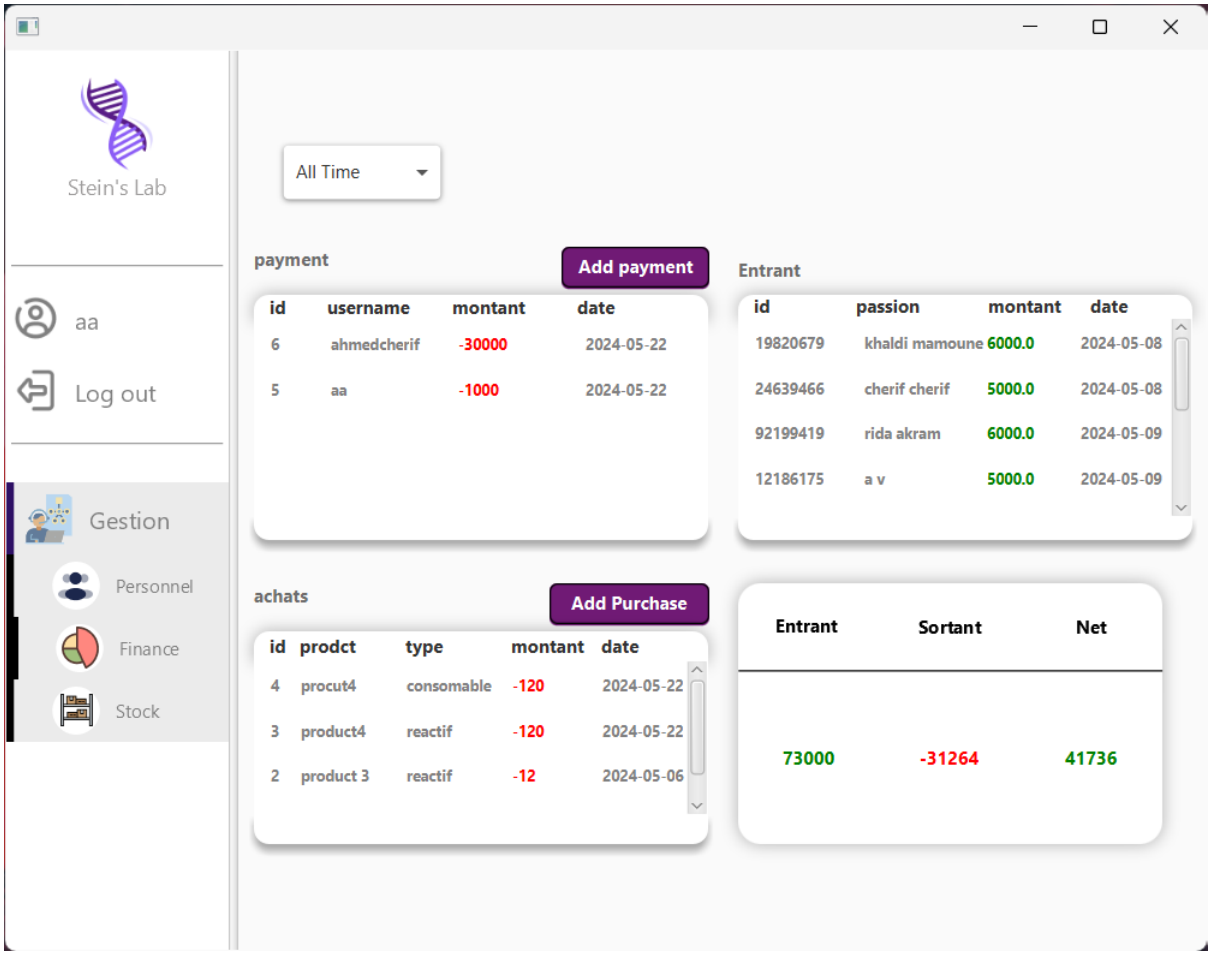

Figure 27: Accounting Interface

Sample Management Interface

The sample management interface allows the biologist to manage the sample test results of patients. The biologist can add or modify a sample test result. The biologist can also search for available samples.

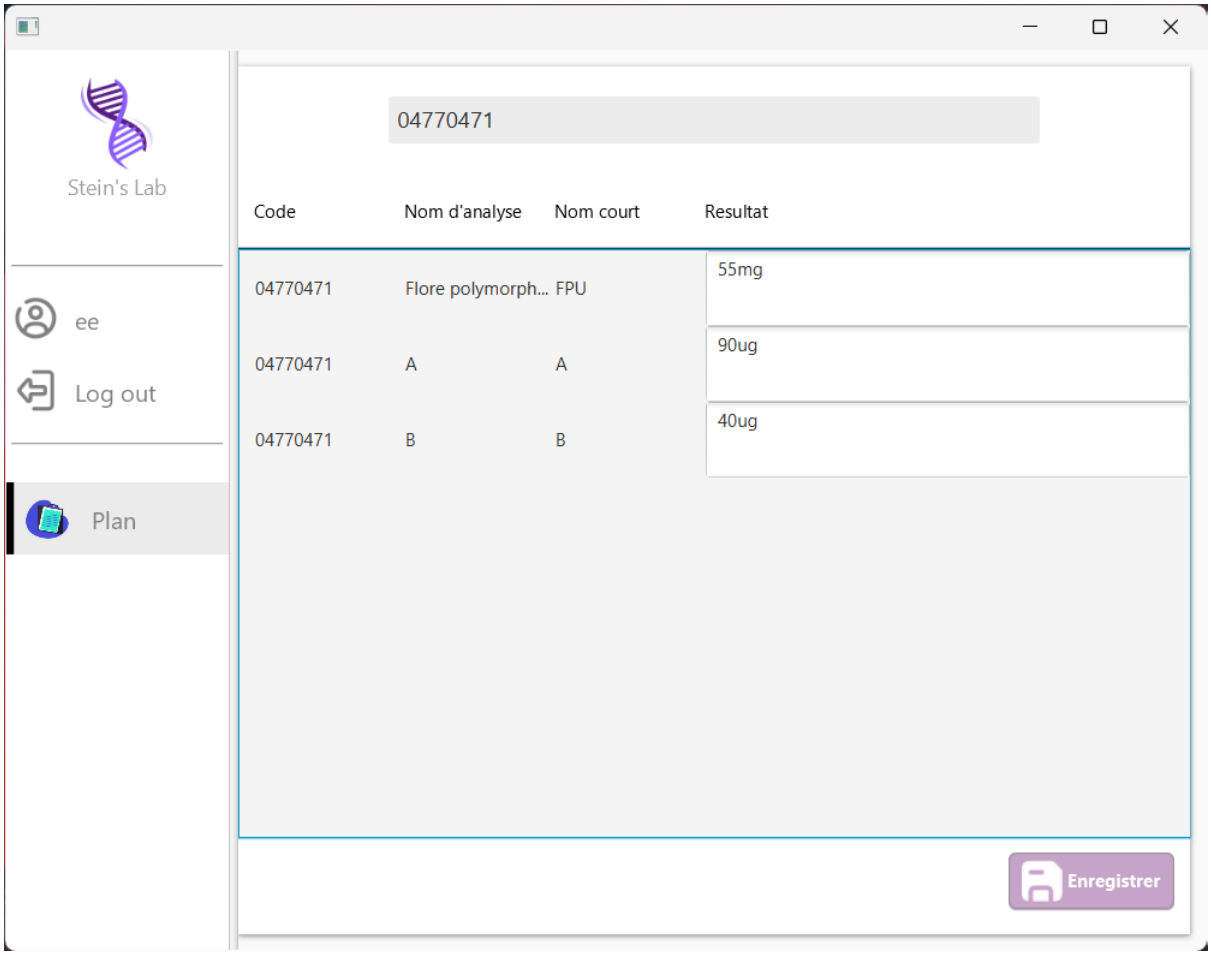

Figure 28: Sample Management Interface

Results Validity Management Interface

The results validity management interface allows the doctor to view the results of each patient's test and choose to validate those results for final delivery or not.

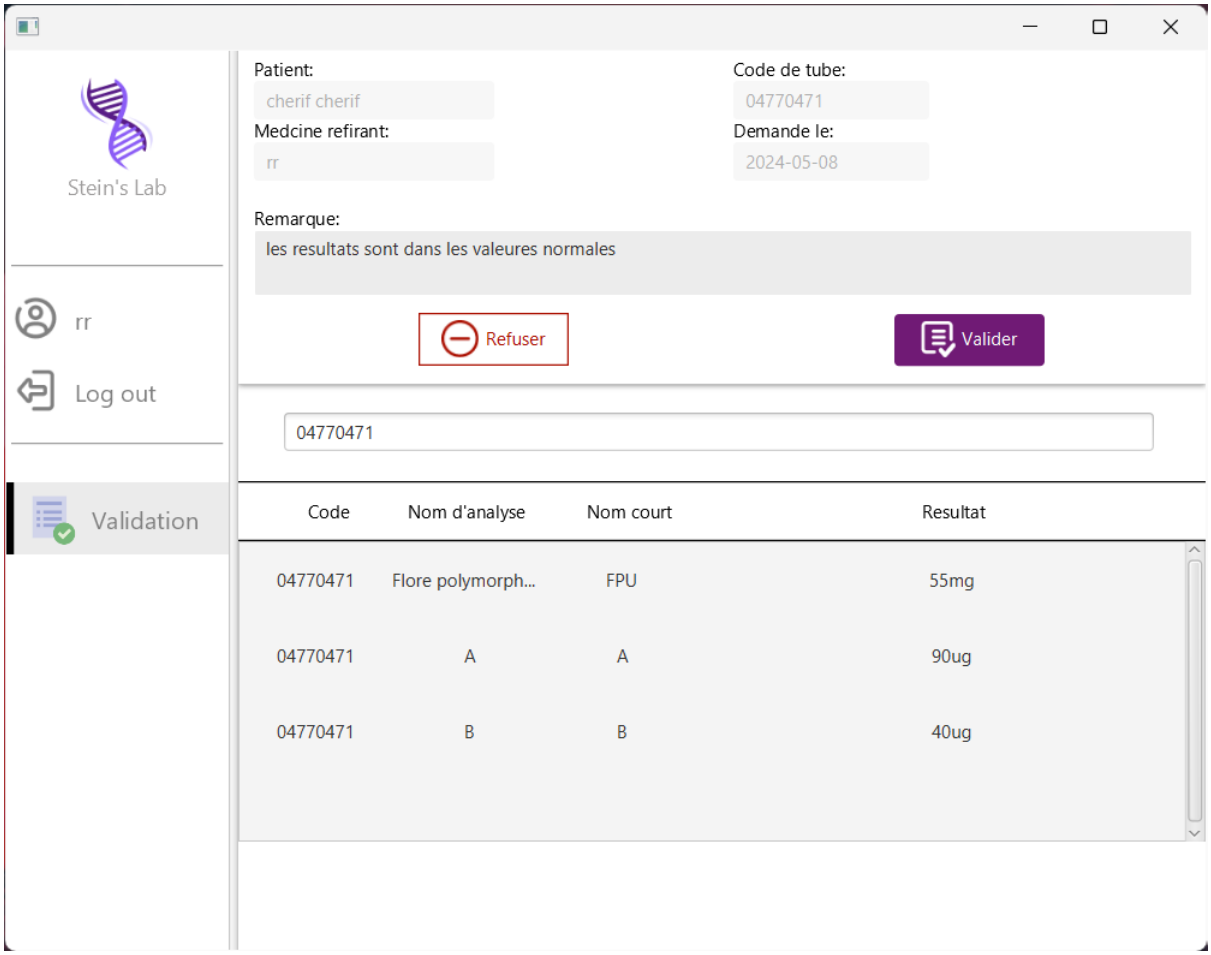

Figure 29: Results Validity Management Interface

Patients & Medical Documents Management Interface

The patients & medical documents management interface allows the receptionist to store and manipulate patients data. The receptionist can add new patients and modify or delete existing ones. The receptionist can also search for available patients. The interface also allows the receptionist to manage medical documents and relevant information for each patient.

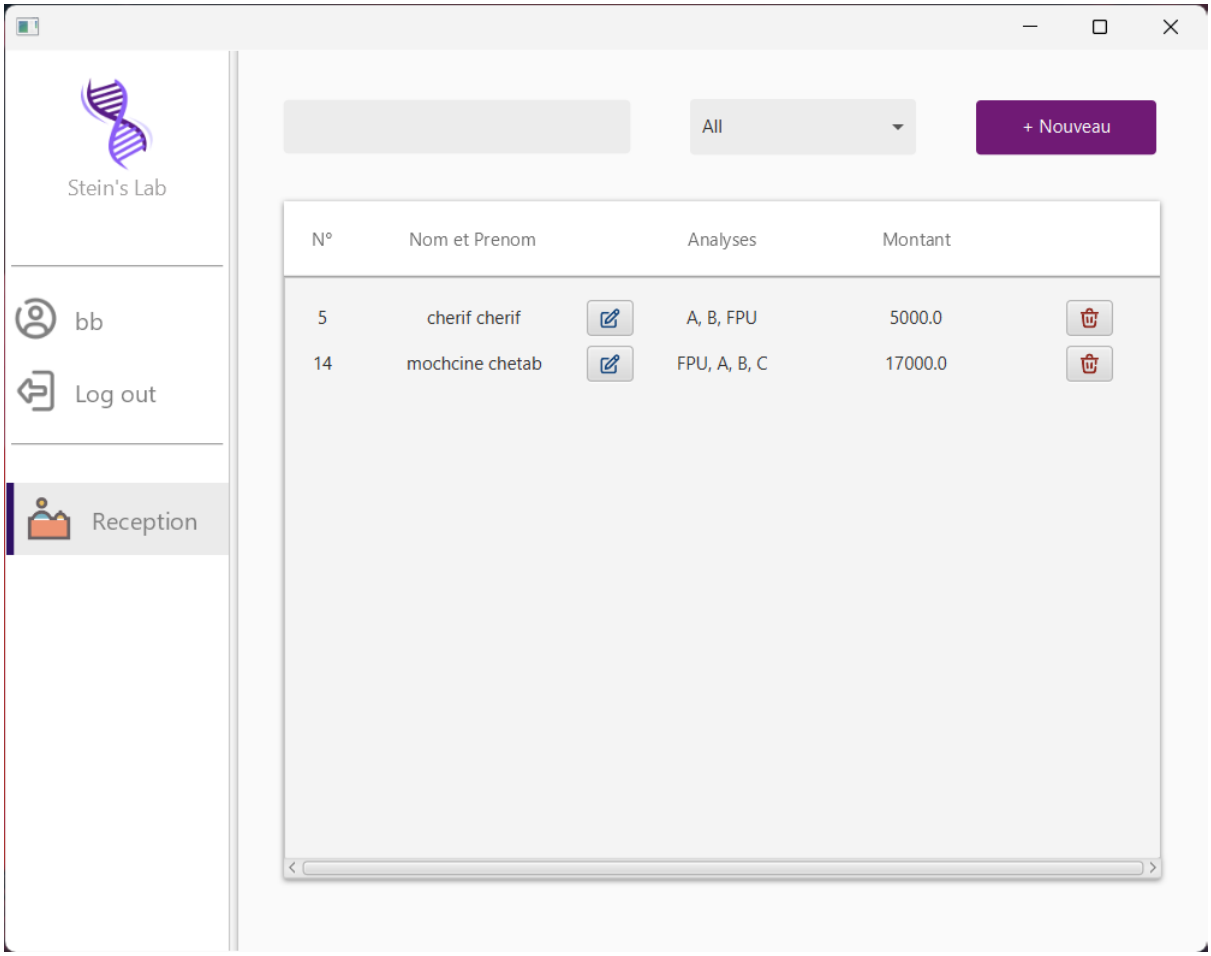

Figure 30: Patients Data Management Interface

6 Conclusion

In this report, we have presented the development of an application for the management of a Medical Analysis Laboratory. We have described the purpose of the application, the existing solutions and their shortcomings, the requirements analysis, the application design, the application implementation, and the project presentation.

The realization of this project required good collaboration between our team members, as well as using best practices for software development. Starting from requirements analysis then design. Implementation required from us that we familiarize ourselves with many new technologies and methodologies.

We believe that our work will be of great benefit to the medical analysis laboratories that are looking for a reliable and easy-to-use software solution to manage their daily workflow.

We focused on bringing a simple and user friendly option to the market, as we noticed that most of the existing solutions are overcomplicated and lack attention to the user experience and user interface.

We hope that our application will help laboratories to improve their efficiency and productivity, and we look forward to receiving feedback from users to further improve our application.

7 References

- https://en.wikipedia.org/wiki/Unified_Modeling_Language
- https://en.wikipedia.org/wiki/IntelliJ_IDEA
- https://en.wikipedia.org/wiki/XAMPP
- https://en.wikipedia.org/wiki/Figma
- https://en.wikipedia.org/wiki/Java_(programming_language)
- https://en.wikipedia.org/wiki/JavaFX
- https://en.wikipedia.org/wiki/MySQL
- https://en.wikipedia.org/wiki/Relational_model
- https://en.wikipedia.org/wiki/Software_design