People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of

the Requirements for the Degree of

MASTER

In **Electronics**

Option: **Computer Engineering**

Title:

DESIGN AND IMPLEMENTATION OF A PEDIATRIC MANAGEMENT SOFTWARE SYSTEM

Presented by:

BELHADJ Mohamed Aymen

KACIMI Hidayete

Supervisor:

Dr. NAMANE R.

Registration number :……./2022

Abstract

As the number of patients increases over time, managing their medical records using the traditional methods became inconvenient for doctors. Consequently, this way of management leads to a waste of time and effort, as well as patients' information is unsecure. The aim of this project is to design and implement an interactive, efficient, and friendly used pediatric software system that helps pediatricians in managing their cabinet. This software application helps doctors to manage their offices in a more modern and easy way with a given number of functionalities that provide a complete patient’s medical record and allow proper diagnosis and treatment since the patient’s history is secured and organized in a database. The medical office of Doctor M.Z. SIDI SAID, a pediatrician working in Boumerdes, is considered as case study.

Dedications

I dedicate this work to my family: my beloved parents, my brother, and my sister who always believed in me and supported me during the hard times.

To my dear friends who have always been there for me. To my best friend who helped me get through rough times. To Araboui's family for their infinite support and especially little Lokmane who always looks up to me.

Lastly, a special thank you to my partner Aymen for his patience and his hard work through the making of this project.

Hidayete

This work is dedicated to:

The sake of Allah, my Creator and my Master,

My great teacher and messenger, Mohammed (May Allah bless and grant him), who taught us the purpose of life,

My great parents, who never stop giving of themselves in countless ways,

My beloved brother and sister; particularly my dearest brother, Mouad, who stands by me,

My friends who encourage and support me,

All the people in my life who touched my heart.

Aymen

Acknowledgements

Praise to God, the supreme for leading us during all these years of our study path, and for giving us the courage and the strength to realize this dissertation successfully, despite all the difficulties, Alhamdulillah.

Our deep and sincere gratitude goes to our teacher and supervisor Dr. Namane who followed this project with great interest and guided us throughout it. We would like also to thank him for his sincerity and his endless motivation that helped us fulfill this work.

Secondly, we are extremely grateful for our teachers who helped us reach this level of knowledge, especially Dr. Khalifa for his valuable course about databases and Dr. Zitouni for his Object-Oriented programming course.

To all the people who contributed in the making of this project, and to all the people who helped us throughout this journey; find here an expression of our deep gratitude.

Thank you

Table of Contents

List of Tables

List of Figures

General introduction

Management information system (MIS) is a computer system that is responsible for information storage, handling, processing and retrieval of data in order to help an organization perform its core function. The role of an MIS can never be overestimated due to the significant advantages it offers, this includes improving the quality of decision making, reducing the paperwork and helping organizations achieve high levels of efficiency which leads to a better performance.

Many doctors' offices still use traditional ways to manage their patients' data. To illustrate, the doctor must write the patient's information on a paper file and label it with an index that has the patient's name; all this work leads to wasting the doctor's time and effort. In addition, these files are stored in a cabinet at the doctor's office which does not assure the security of patients' data.

In light of this situation, our project's main objective is to design and implement an interactive desktop application that ensures the security of the patient's data, facilitates the doctor's job by optimizing time and space, improves the quality of data control and provides flexibility and accessibility to the patients' information easily.

This report is divided into three chapters: The first chapter introduces an overview of desktop applications and the features they provide when being used in the medical field; in addition, the tools and technologies used in the implementation of our desktop application are presented. The second chapter describes the design of our application using different UML diagrams in addition to some generalities about databases along with the application's database schema. In the third chapter, the different interfaces of our application are illustrated by showing multiple scenarios. The report terminates with a general conclusion that summarizes the presented work and provides some suggestions for future works.

Chapter One: Theoretical background

1.1. Introduction

During the recent decade, the use of computers has grown rapidly, causing the traditional ways of managing medical offices to be unsuitable for the new era. Managing the patient information in the healthcare field has been developed and improved through different manners such as database servers, web applications, computer applications, etc.

In this chapter, we will present an overview about desktop applications, their advantages in managing a medical office and the tools we used to implement this application.

1.2. Desktop applications overview

Desktop applications fall into the category of management information systems; these systems are increasingly being used for information storage, handling, processing and retrieval of data for improving the services provided by any organization. They allow quick decision making for overall functional improvement. [1]

1.3. Why use a desktop application?

Many organizations and individuals get confused with the vast options to choose between to manage their information, especially between web and desktop applications. As it is known that web based applications have their advantages such as the cross platform compatibility, but desktop applications are more reliable in terms of privacy since online access is not necessary which protects the machine from man-in-the-middle and other threats in online mode. On the other hand, the performance of desktop applications for complex calculations is more optimized which can be both time and cost efficient.

1.4. Desktop applications in the medical field

Desktop applications can serve as a convenient tool for helping doctors such as better diagnoses, selection of dosage of medicines based on the patient's indicators, accessing the medical files and information of the patients, managing appointments and more functionalities like general statistics on the patient’s data .In addition, desktop application provide a more efficient way to store patient’s data comparing to using traditional ways such as paper files, and also a more secure way since that data is stored in the user’s computer. However, these programs do not intend to replace the doctor, and only to optimize his work.

1.5. Tools used

1.5.1. C# programming language

The C# programming language (pronounced “see sharp”) is used for many kinds of applications, including websites, cloud-based systems, IoT devices, machine learning, desktop applications, embedded controllers, mobile apps, games, and command-line utilities. C#, along with the supporting runtime, libraries, and tools known collectively as .NET, has been center stage for Windows developers for almost two decades, but in recent years, it has also made inroads into other platforms. [2]

1.5.2. Microsoft office access database

1.5.2.1. Definition

Microsoft Access is a database management system (DBMS) from Microsoft that combines the relational Access Database Engine (ACE) with a graphical user interface and software-development tools. Microsoft Access stores data in its own format based on the Access Database Engine (formerly Jet Database Engine). It can also import or link directly to data stored in other applications and databases. [3]

1.5.2.2. Microsoft office access advantages

Microsoft Office Access offers numerous benefits that make it extremely useful when developing desktop applications. Some of these benefits include:

- **Cost of development:** One of the reasons why Microsoft Access is consistently the database of choice is that it is a less expensive alternative to larger database systems, such as Oracle or SQL (Structured Query Language) Server, which require a tremendous amount of set up and maintenance costs.
- **Software integration:** Since Access is a Microsoft Office product, it has been designed to integrate well with other products in the Microsoft Office Suite, Access is such a widely used database system that other software manufacturers are more likely to provide the ability to interface directly to Access than any other desktop database system.

- **ODBC (Open Database Connectivity) compliant:** Access has the ability to use data stored in Access/Jet, Microsoft SQL Server, Oracle and other ODBC compliant data containers including MySQL and PostgreSQL. Software developers and data architects have used Access to easily develop application software.
- **Legacy data:** Access has the ability to easily import many data formats so that existing data is not lost. This feature can not only save 100′s of hours of input time, but also eliminate potential human input error.
- **Distribution:** One of the benefits of using Microsoft Access is its Jet Database format, which contains not only the application but also the data in one file. The ability to have the application and data in one file makes it extremely convenient to distribute the entire application to various users who can run the application in disconnected environments.
- **Unique identifier:** Each piece of information is assigned a unique identifier. This is extremely important since it controls that each piece of information is only entered once and eliminates human and duplication errors. [4]

1.5.3. Visual Studio IDE (Integrated Development Environment)

The Visual Studio IDE is a software program made by Microsoft for developers to write and edit their codes. Its user interface is used for software development to edit, debug, and build code. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, designer for building GUI applications, web designer, class designer, and database schema designer. [5]

The figure below shows Visual Studio interface with an open project that shows key windows and their functionality:

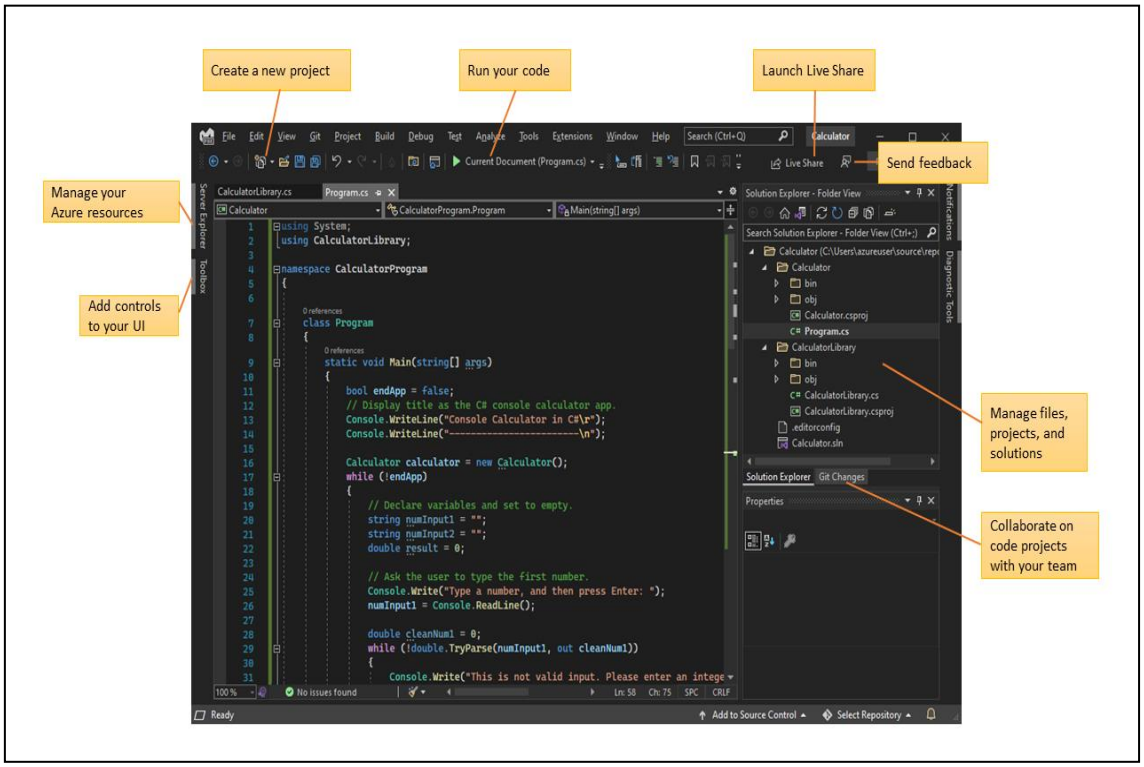

Fig 1. 1 Key windows in Visual Studio

1.5.4. .NET Framework

1.5.4.1. Definition

.NET framework is a software development environment designed and supported by Microsoft in 2002 and is used for compiling and executing programs aiming to build different types of applications. Supporting more than 60 programming languages 11 of which are designed and developed by Microsoft, .NET's primary languages are C#, VB.NET and ASP.NET which are used for developing form-based and web-based applications. Due to the features it offers, developers prefer the .NET framework as it relieves them from handling security operations, active memory management, and other low-level efforts.

1.5.4.2. Architecture and components

.NET framework consists of two basic components: Common Language Runtime (CLR) and Framework Class Library (FCL) in addition to more components that were added by Microsoft when updating .NET. The components are shown in the figure below:

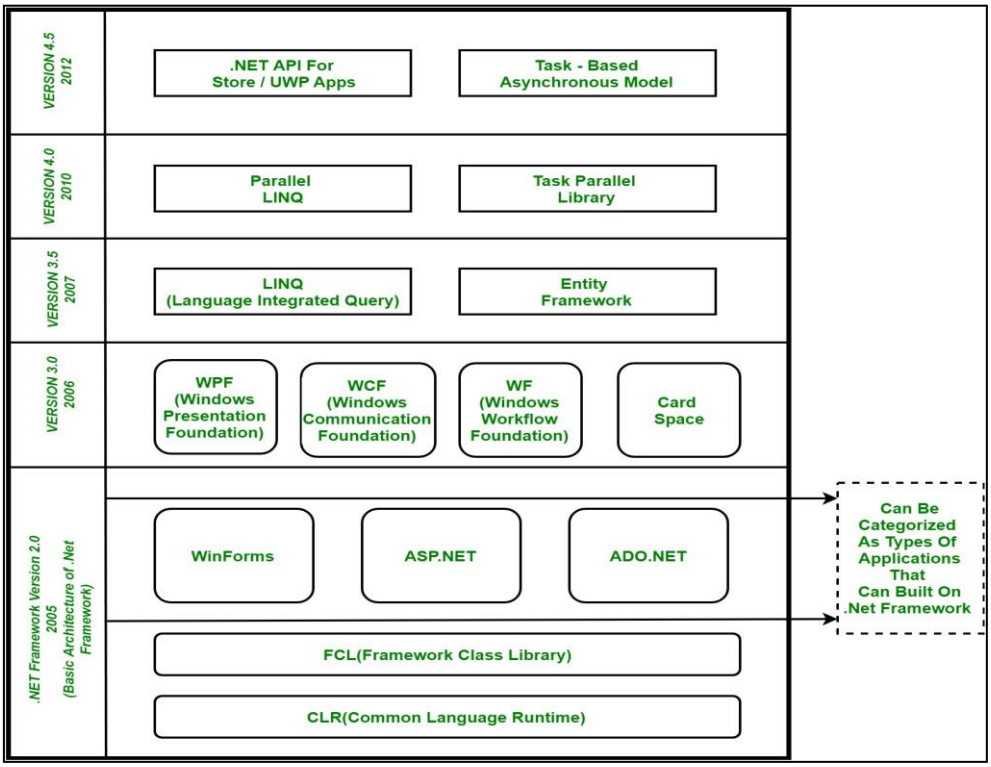

Fig 1. 2 Basic Architecture and Component Stack of .NET Framework

- **CLR (Common Language Runtime):** A run-time environment used to execute the programs written in any language supported by .NET framework such as C# in the case of developing desktop applications.
- **FCL (Framework Class Library):** Includes a significant number of class libraries, interfaces and data types in order to allow access to system functionality.
- **Windows Forms (WinForms):** Windows Forms is a user interface (UI) framework for building Windows desktop applications. It provides one of the most productive ways to create desktop apps based on the visual designer provided in Visual Studio. Functionality such as drag-and-drop placement of visual controls makes it easy to build applications.

  Windows Forms contains a variety of controls that can be added to forms such as controls that display text boxes, buttons, drop-down boxes, radio buttons, webpages and even charts. It also supports creating own custom controls using the User Control class. [6]
- **ADO.NET:** Mainly used to provide access and communicate with the data sources exposed through OLE DB (Object Linking and Embedding Database) and ODBC (Open Database Connectivity). After connecting to the database, ADO.NET uses data providers for manipulating data, executing commands and retrieving results. The following ADO.NET components were used in building our application:

- **Dataset:** Also known as disconnected data access method, it consists of a collection of data tables used to fetch data to be used in a running program without having to interact with the data source. A dataset contains one or more tables; each table can contain one or more columns and rows. If a dataset contains two or more tables, it can also define the relationship between those two tables. Although a dataset is structured much like a relational database, it is important to realize that each table in a dataset corresponds to the result set that is returned from a SELECT statement, not necessarily to an actual table in a database. For example, a SELECT statement may join data from several tables in a database to produce a single result set. In this case, the table in the dataset would represent data from each of the tables involved in the join. [7] The figure below illustrates the basic organization of an ADO.NET dataset:

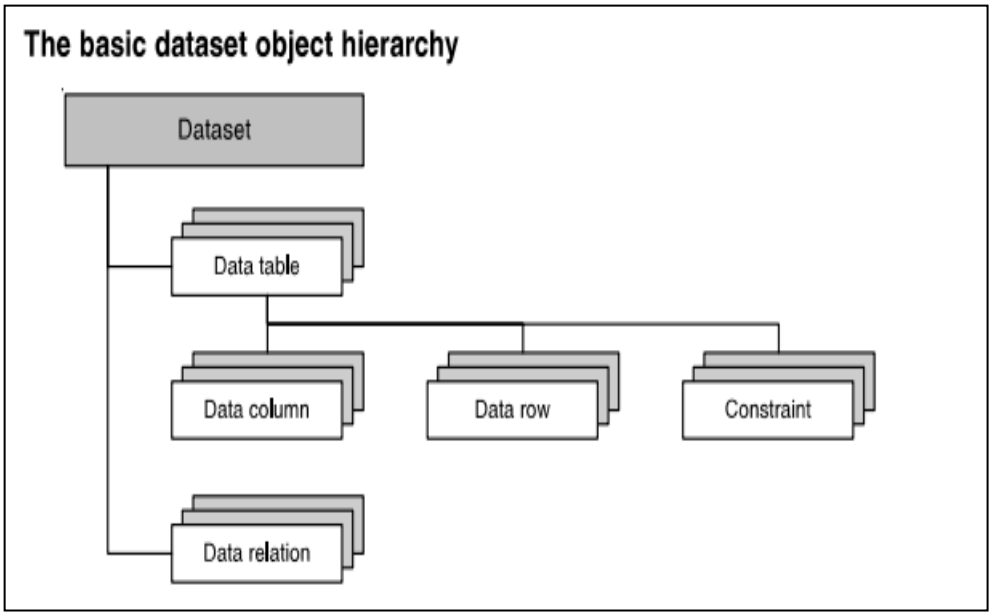

Fig 1. 3 The basic dataset object hierarchy

- **TableAdapter:** A TableAdapter component fills a dataset with data from the database, based on one or more queries or stored specified procedures. TableAdapters can also perform adds, updates, and deletes on the database to persist changes made to the dataset. They connect to the database, run queries or stored procedures, and either return a new data table or fill an existing DataTable with the returned data. TableAdapters can also send updated data from the application back to the database. [8]

- **Binding source:** The BindingSource component is designed to simplify the process of binding controls to an underlying data source. It acts as both a conduit and a data source for other controls to bind to. It provides an abstraction of the form's data connection while passing through commands to the underlying list of data. Additionally, data can be added directly to it, so that the component itself functions as a data source. [9]

1.5.5. Bunifu Framework and FontAwesome tools:

These consist of a collection of tools that help developers to create modern and interactive interfaces for application with a wide variety of customization that allows them to create unique and stylish designs. Both Bunifu framework and FontAwesome tools support WinForms for C# and offered a great help in designing our application.

1.6. Conclusion:

In this chapter, we have introduced an overview of desktop applications and their use in the medical field. Furthermore, we have detailed the used tools to build a given desktop application.

Chapter Two: System Design

2.1. Introduction

System design is a major step in building our desktop application as it provides better understanding of the system and its behavior. In this chapter we will introduce the unified modeling language (UML) and its different diagram types used in designing our application.

2.2. Unified Modeling Language

UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML provides a standard way to write a system's blueprints, covering conceptual things, such as business processes and system functions, as well as concrete things, such as classes written in a specific programming language, database schemas, and reusable software components. UML defines a number of diagrams that fall into two groups: behavioral diagrams and structural diagrams. [10]

- **Structural (or Static) view**: emphasizes the static structure of the system using objects, attributes, operations and relationships. It includes class diagrams and composite structure diagrams. [11]
- **Behavioral (or Dynamic) view**: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams, and state machine diagrams. [11]

The following figure illustrates the types of UML diagrams:

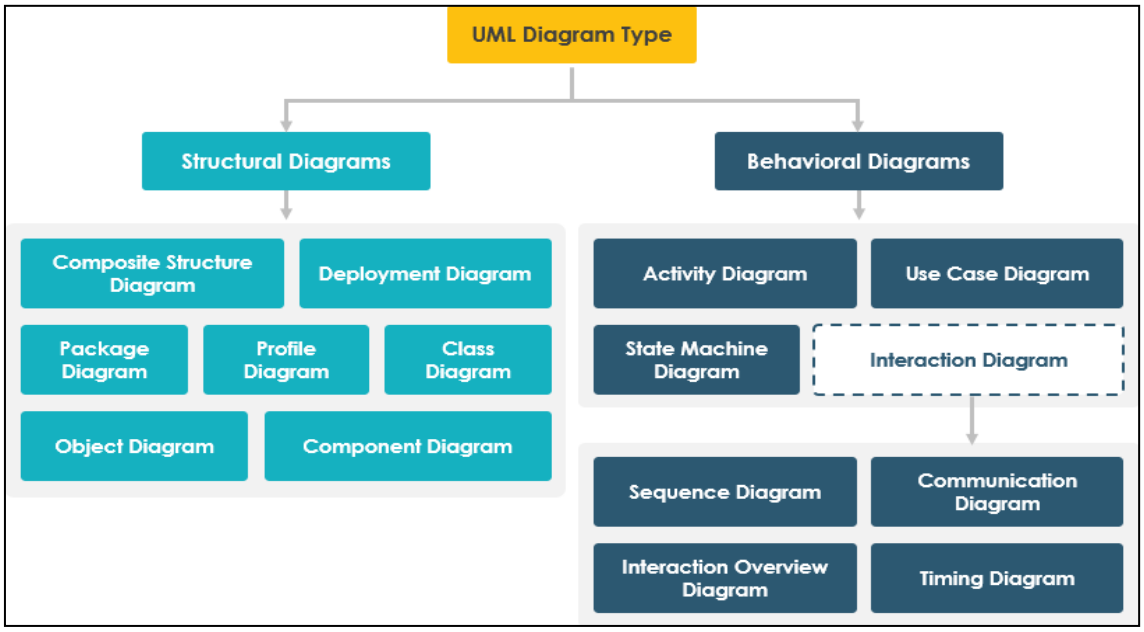

Fig 2. 1 Types of UML diagrams

2.3. Use Case Diagram

2.3.1. Definition

Use case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally. [12]

Generally, a use case diagram consists of four major components described as follows:

a. **Actors:** An actor portrays any entity (or entities) that perform certain roles in a given system. The different roles the actor represents are the actual business roles of users in a given system. An actor in a use case diagram interacts with a use case. [13]
   For our application, we have two main actors: the doctor and the receptionist.

b. **System boundary:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario. It defines the scope of what a system will be. [14]

c. **Relationships:** The use case diagram has two types of relationships: one between two use cases which models the dependency between those use cases, these include:

   Include: Described by a dotted arrow labeled with the keyword «include». The «include» relationship declares that the use case at the head of the dotted arrow completely reuses all of the steps from the case being included. [15]

   Extend: It is a directed relationship that specifies how and when the behavior defined is supplementary (optional), extending use case can be inserted into the behavior defined in the extended use case. Extend relationship is shown as a

dashed line with an open arrowhead directed from the extending use case to the extended (base) use case. The arrow is labeled with the keyword «extend». [16]

The second relation is between actors and use cases; it is known as association and it indicates that the actor interacts or communicates with the use case. It is shown as a straight line.

d. Use case: Defines a sequence of interactions between the actor and the system. It is represented by an ellipse with a text describing the function. For our application, we defined the following use cases:

i. **Authentication:** Requires login to access the application using a valid username and password.

ii. **Patient management:** Allows the doctor to search for a specific patient, add a new patient, edit the information of an existing patient or delete a patient.

iii. **Appointments:** Used by the receptionist to schedule appointments and label them as either processed or canceled.

iv. **Examination:** Allows the doctor to add a new examination, this includes selecting reason of visit, symptoms, diagnosis or complementary examination from the list given or adding a new item.

v. **Treatment:** Allows the doctor to prescribe the necessary quantity and dosage of the medicine in addition to the possibility of printing the prescription.

vi. **Statistics:** Displays statistics and graphs as needed, the doctor has to select the items and relations needed to be displayed and the graph type.

vii. **Tools:** Used by the doctor to manage accounts and give limited permission to certain users using a certain username and password.

The table below shows each actor and their corresponding use cases:

Table 2. 1 Actors and their use cases

Actor	Use case
Doctor	-Authentication -Patient management -Examination -Treatment -Statistics -Tools
Receptionist	-Authentication -Appointments

2.3.2. Application's use case diagrams

2.3.2.1. Receptionist's use case diagram

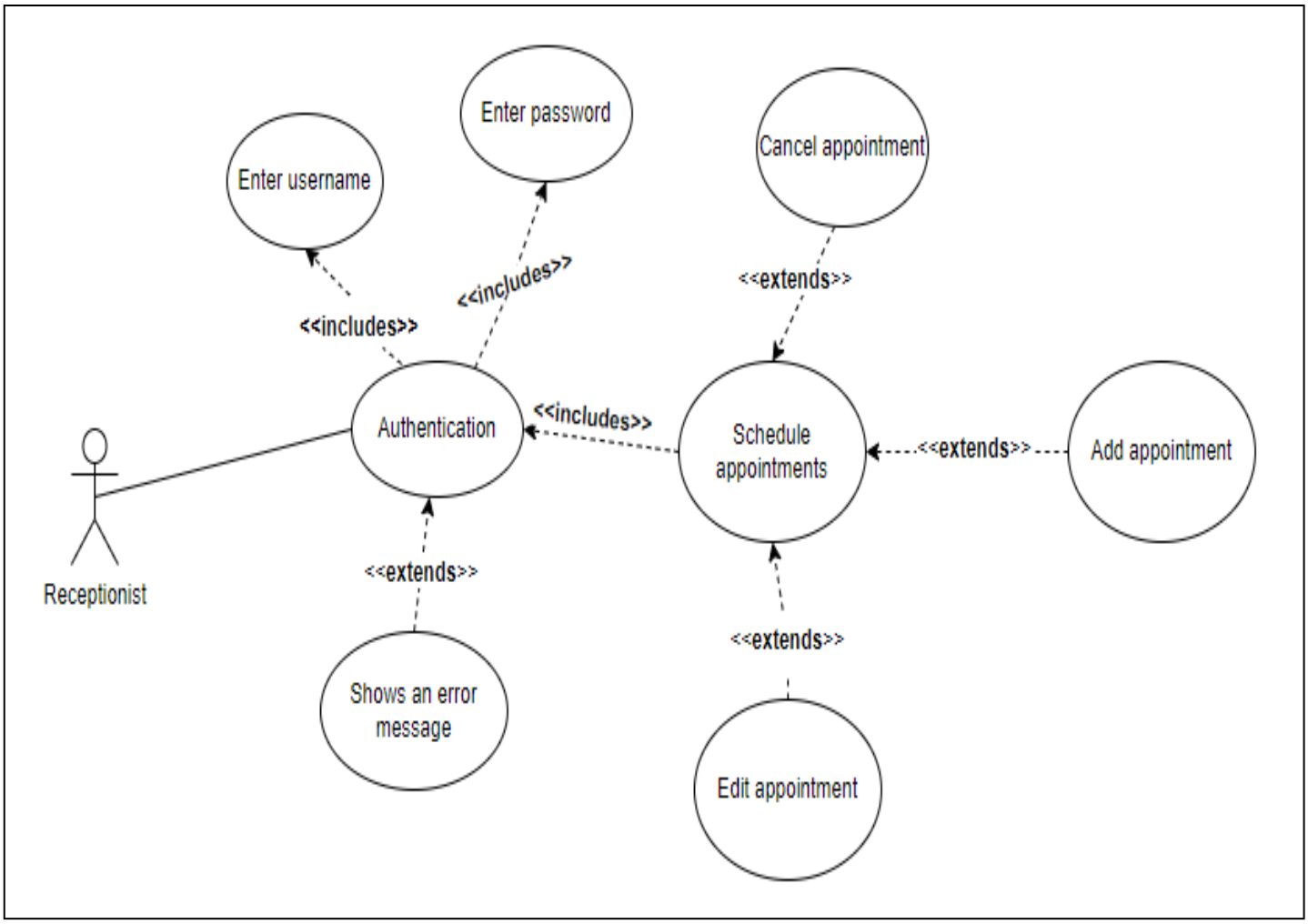

Fig 2. 2 Receptionist's use case diagram

2.3.2.2. Doctor's use case diagram

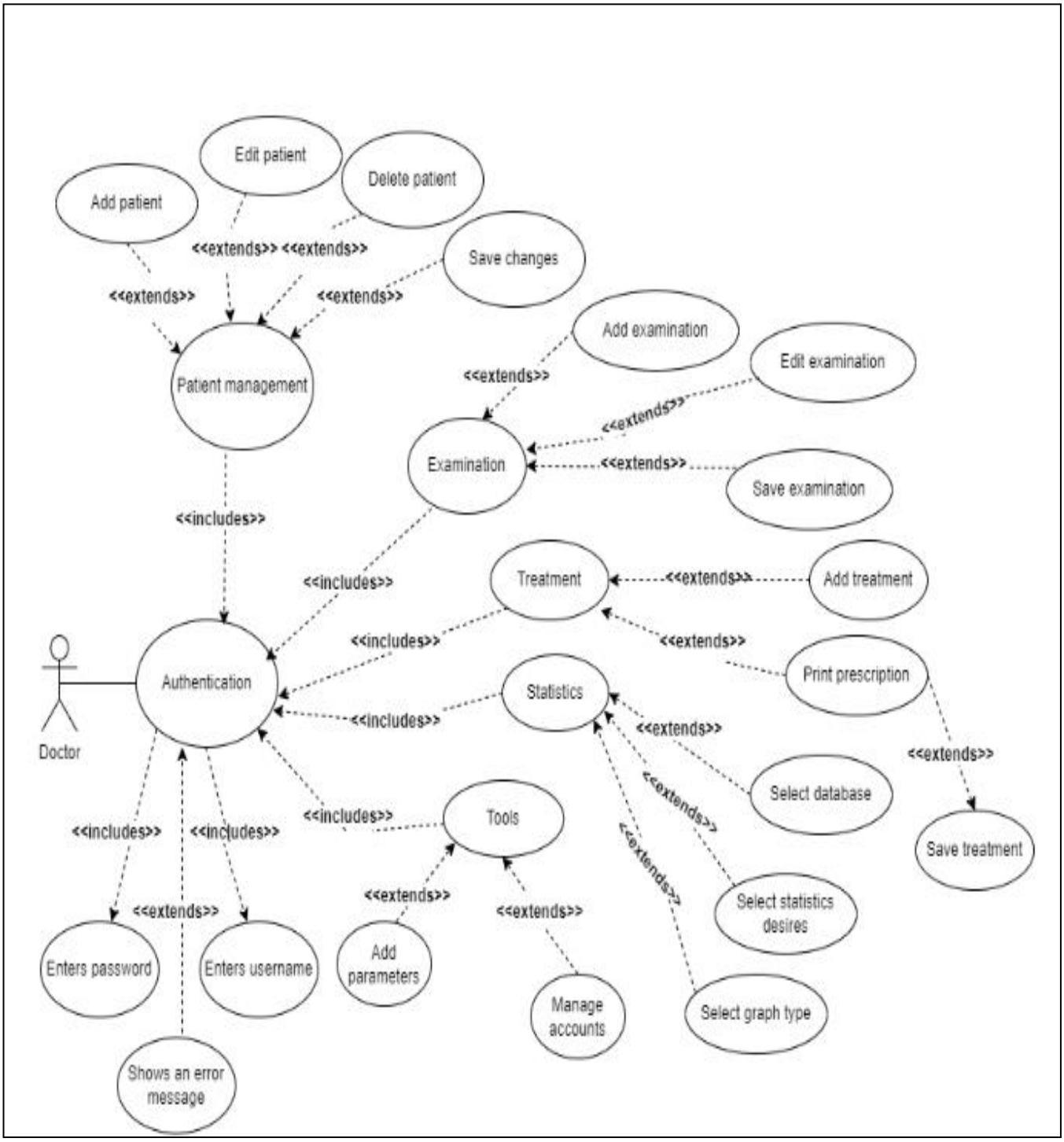

Fig 2. 3 Doctor's use case diagram

2.3.3. Textual description of use cases

2.3.3.1. Receptionist's use case diagram textual description

2.3.3.1.1. Authentication use case

Table 2. 2 Receptionist's authentication use case textual description

Use case name	Authentication
Actor	Receptionist
Object	Login and have access to the application
Precondition	None
Scenario	1- The receptionist launches the application. 2- The application requests a username and password. 3- The user enters his/her username and password. 4- The system sends an authentication query to the server to check if the information entered is correct. 5- The server verifies the query and sends favorable answer. 6- The receptionist accesses the application.
Alternative	If the receptionist enters a wrong username or password, an error message is displayed (return to 2).

2.3.3.1.2. Schedule appointments use case

Table 2. 3 Receptionist's schedule appointments use case textual description

Use case name	Schedule appointments
Actor	Receptionist
Object	Add, edit or cancel appointments
Precondition	Authentication
Scenario	1- The receptionist selects "Appointments" page. 2- The receptionist adds a new appointment. 3- The receptionist edits an existing appointment.
Alternative	The receptionist cancels the appointment if the patient doesn't arrive or label it as processed as soon as the patient enters the doctor's office.

2.3.3.2.Doctor’s use case diagram textual description

2.3.3.2.1. Authentication use case

Table 2. 4 Doctor’s authentication use case textual description

Use case name	Authentication
Actor	Doctor
Object	Login and have access to the application
Precondition	None
Scenario	1- The receptionist launches the application. 2- The application requests a username and password. 3- The doctor enters his username and password and clicks “Login”. 4- The system sends an authentication query to the server to check if the information entered is correct. 5- The server verifies the query and sends favorable answer. 6- The doctor accesses the application.
Alternative	If the doctor enters a wrong username or password, an error message is displayed (return to 2).

2.3.3.2.2. Patient management use case

Table 2. 5 Doctor’s patient management use case textual description

Use case name	Patient management
Actor	Doctor
Object	Add a new patient, view, delete or edit the information of an existing patient.
Precondition	Authentication
Scenario	1-The doctor selects “Patients” page. 2-The doctor chooses an appropriate filter for his search. 3- The system checks the existence of the patient and displays the result in a list. 4- The doctor selects a patient from the list and clicks the edit button.

	5- Civil status page of that patient appears 6-The doctor fill boxes with information related to the patient 7-The doctor moves to Anamnesis page 8- The doctor edits information related to the patient 9-The doctor clicks “Save” button to save changes
Alternative	If the patient does not exist, an error message is displayed (return to 2).

2.3.3.2.3. Examination use case

Table 2. 6 Doctor’s examination use case textual description

Use case name	Examination
Actor	Doctor
Object	Add a new examination, includes selecting symptoms and giving a proper diagnosis.
Precondition	Patient management
Scenario	1-The doctor select “Examination” page. 2-The doctor selects a reason of visit from a stored list. 3-The doctor selects the symptoms from a stored list. 4-The doctor selects a disease from a stored list. 5-The doctor selects a complementary examination from a stored list. 6-The doctor clicks “Save” button to save changes.
Alternative	If one of the items to be selected in the scenario above does not appear in the stored list, a message appears asking the doctor if he wants to add a new item.

2.3.3.2.4. Treatment use case

Table 2. 7 Doctor’s treatment use case textual description

Use case name	Treatment
Actor	Doctor
Object	Prescribe the appropriate medicine to the patient and print the prescription.
Precondition	Examination

Scenario	1-The doctor selects “Treatment” page. 2-The doctor selects a medicine from the medicine list. 3-The doctor selects a quantity and dosage from a stored list 4-The doctor clicks “Print” button to print prescription and save changes to the examination history page.
Alternative	If one of the items to be selected in the scenario above does not appear in the stored list, a message appears asking the doctor if he wants to add a new item.

2.3.3.2.5. Statistics use case

Table 2. 8 Doctor’s statistics use case textual description

Use case name	Statistics
Actor	Doctor
Object	Display different statistics according to the doctor’s choice
Precondition	Authentication
Scenario	1-The doctor selects “Statistics” page. 2-The doctor selects a specific database from which the data will be retrieved. 3-The doctor selects statistics desired. 4-The doctor selects a specific graph type from a given list. 5-The doctor clicks “Render” button. 6-The system displays the desired statistics.
Alternative	None

2.3.3.2.6. Tools use case

Table 2. 9 Doctor’s tools use case textual description

Use case name	Tools
Actor	Doctor
Object	Allows the doctor to create new account, edit the old ones or add needed parameters.
Precondition	Authentication
Scenario	1-The doctor selects “Tools” page.

	2-The doctor clicks “Manage accounts” 3-The doctor edits the information of an existing account. 4-The doctor clicks “Save changes” button. 5-The doctor adds a new account. 6-The doctor clicks “Save changes” button. 7- The doctor deletes an existing account. 8-The doctor clicks “Save changes” button.
Alternative	The doctor selects “Add parameters” to add new symptoms, diagnoses, letters…etc

2.4. Class Diagram

2.4.1 Definition

A class diagram is a type of diagram and part of a unified modeling language (UML) that defines and provides the overview and structure of a system in terms of classes, attributes and methods, and the relationships between different classes. It is used to illustrate and create a functional diagram of the system classes and serves as a system development resource within the software development life cycle. [17]

2.4.2 Visibility of attributes and operations

A class diagram is defined as a set of classes containing attributes and operations, connected to each other by relations and have conditions of participation. The attributes and operations have different levels of visibility, each is described as follows:

- **Public:** Represented by ‘+’, denotes the element that is visible to all elements that can access the contents of the namespace that owns it.
- **Private:** Represented by ‘-’, denotes the element that is only visible inside the namespace that owns it.
- **Protected:** Represented by ‘#’, denotes the element that is visible to elements that have a generalization relationship to the namespace that owns it.

- **Package**: Represented by '~', a package element is owned by a namespace that is not a package, and is visible to elements that are in the same package as its owning namespace. [18]

2.4.3 Relationships between classes:

A class may be involved in one or more relationships with other classes. The relationships used in our class diagram are cited in the following table with their corresponding graphical representation:

Table 2. 10 Class diagram relationships

Relationship	Graphical representation
Simple association: Indicates that instances of one class are connected to instances of another class.	
Aggregation: A special type of association which indicates that a class is a part of another class. A number of instances of class 2 denoted by * can be associated with class 1.	
Composition: A special type of aggregation which indicates that a certain class cannot stand by itself and only exists if the other class does.	
Inheritance: Indicates that the child class receives all the attributes, operations, and relationships that are defined in the parent.	

2.4.4 Multiplicity

Multiplicity stands for how many objects of each class take part in the relationships. It can be expressed as follows:

- **1** Exactly one.
- **0..1** Zero or one.
- ***** Many.
- **0..*** Zero or many.
- **1..*** One or many.

2.4.5 Application class description

Before drawing our class diagram, we need to define all the classes and add attributes and operations for those classes. Each class has to be of the following form:

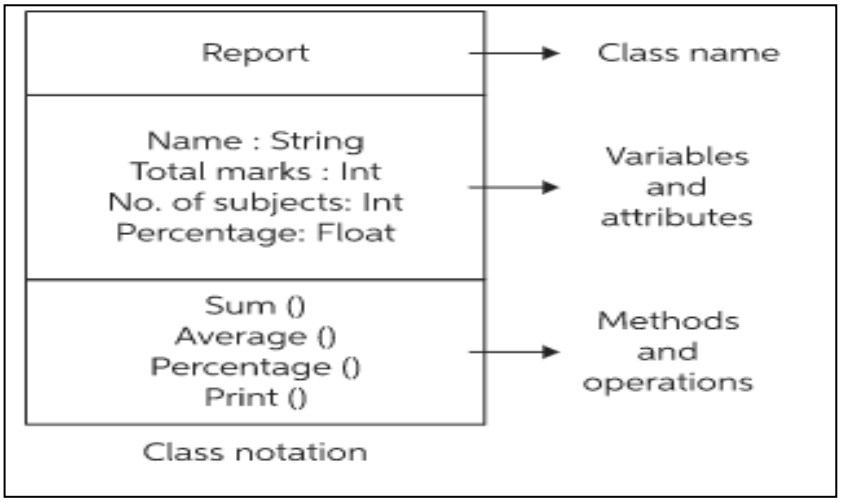

Fig 2. 4 Class notation

All the classes used in our class diagram in addition to their corresponding attributes and operations are listed in the following tables:

Table 2. 11 User and Patient class description

Class name: User		Class name: Patient	
Attributes	Type	Attributes	Type
u_id	Integer	p_id	Integer
u_username	String	p_fname	String
u_password	String	p_lname	String
u_name	String	p_gender	String
u_permission	Integer	p_birthdate	DateTime
		p_registerdate	DateTime
		p_fathername	String
		p_mothername	String
		p_address	String
		p_phone	Integer
		p_childnbrm	Integer
		p_childnbrf	Integer
Methods and operations		Methods and operations	

addUser()	addPatient()
editUser()	editPatient()
deleteUser()	deletePatient()
saveUser()	selectPatient()
verifyLogin()	searchPatient()

Table 2. 12 Anamnesis and Examination class description

Class name: Anamnesis		Class name: Examination	
Attributes	Type	Attribute	Type
anam_id	Integer	exam_id	Integer
anam_date	DateTime	exam_date	DateTime
anam_BCG_P	Boolean	exam_age	Integer
anam_DTCOQ_3M	Boolean	exam_height	Double
anam_DTCOQ_4M	Boolean	exam_pc	Double
anam_DTCOQ_5M	Boolean	exam_weight	Double
anam_DTCOQ_18M	Boolean		
anam_DTA_11A	Boolean		
anam_DTA_16A	Boolean		
anam_ROUGEOLE_9M	Boolean		
anam_DTC_P_R_6A	Boolean		
anam_VitamD_1M	Boolean		
anam_VitamD_6M	Boolean		
anam_TABAC	Boolean		
anam_bLocation	String		
anam_feeding	String		
anam_artificialFeeding	String		
anam_reaInterval	String		
anam_reaTreatment	String		
anam_weight	Double		
anam_birthTime	DateTime		
anam_matInterval	String		
anam_birth	String		
anam_PROPRETE	Boolean		

Methods and operations	Methods and operations
addAnamnesis() editAnamnesis() deleteAnamnesis() saveAnamnesis() selectAnamnesis()	addExamination() editExamination() deleteExamination() saveExamination() selectExamination()

Table 2. 13 Treatment, FatherJob and MotherJob class description

Class name: Treatment		Class name: FatherJob		Class name: MotherJob	
Attributes	Type	Attributes	Type	Attribute	Type
treat_id treat_date	Integer DateTime	Job_id Job_name	Integer String	Job_id Job_name	Integer String
Methods and operations		Methods and operations		Methods and operations	
addTreatment () editTreatment () deleteTreatment () saveTreatment () selectTreatment ()		addFatherJob () editFatherJob () deleteFatherJob () saveFatherJob () selectFatherJob()		addMotherJob () editMotherJob () deleteMotherJob () saveMotherJob () selectMotherJob()	

Table 2. 14 Appointment, Analysis and AnalysisDetails class description

Class name: Appointment		Class name: Analysis		Class name: AnalysisDetails	
Attributes	Type	Attributes	Type	Attribute	Type
app_id app_date	Integer DateTime	analysis_id analysis_name	Integer String	aDetails_id aDetails_name	Integer String
Methods and operations		Methods and operations		Methods and operations	
addAppointment () editAppointment() cancelAppointment () saveAppointment () selectAppointment()		addAnalysis() editAnalysis() deleteAnalysis() saveAnalysis() selectAnalysis()		addAnalysisDetails () editAnalysisDetails() deleteAnalysisDetails() saveAnalysisDetails() selectAnalysisDetails()	

Table 2. 15 Diagnosis, Symptoms and VisitReason class description

Class name: Diagnosis		Class name: Symptoms		Class name: VisitReason	
Attributes	Type	Attributes	Type	Attribute	Type
diag_id diag_name	Integer String	sympt_id sympt_name	Integer String	vReason_id vReason_name	Integer String
Methods and operations		Methods and operations		Methods and operations	
addDiagnosis () editDiagnosis() deleteDiagnosis() saveDiagnosis () selectDiagnosis()		addSymptom() editSymptom() deletSymptom() saveSymptom() selectSymptom()		addVisitReason() editVisitReason() deleteVisitReason() saveVisitReason() selectVisitReason()	

Table 2. 16 Letter, Certificate and Prescription class description

Class name: Letter		Class name: Certificate		Class name: Prescription	
Attributes	Type	Attributes	Type	Attribute	Type
letter_id letter_name	Integer String	certificate_id certficate_name certificate_date	Integer String DateTime	presc_id presc_name presc_volume	Integer String String
Methods and operations		Methods and operations		Methods and operations	
addLetter() editLetter() deleteLetter() saveLetter () selectLetter()		addCertificate() editCertificate() deleteCertificate() saveCertificate() selectCertificate()		addPrescription() editPrescription() deletePrescription() savePrescription() selectPrescription()	

Table 2. 17 Address, State and City class description

Class name: Address		Class name: State		Class name: City	
Attributes	Type	Attributes	Type	Attribute	Type
address_id address_name	Integer String	state_id state_name	Integer String	city_id city_name	Integer String
Methods and operations		Methods and operations		Methods and operations	
addAddress() editAddress() deleteAddress () saveAddress ()		addState() editState() deleteState() saveState()		addCity() editCity() deleteCity() saveCity()	

selectAddress ()	selectState()	selectCity()

2.4.6 Application class diagram

The detailed class diagram of our application is shown in the figure below:

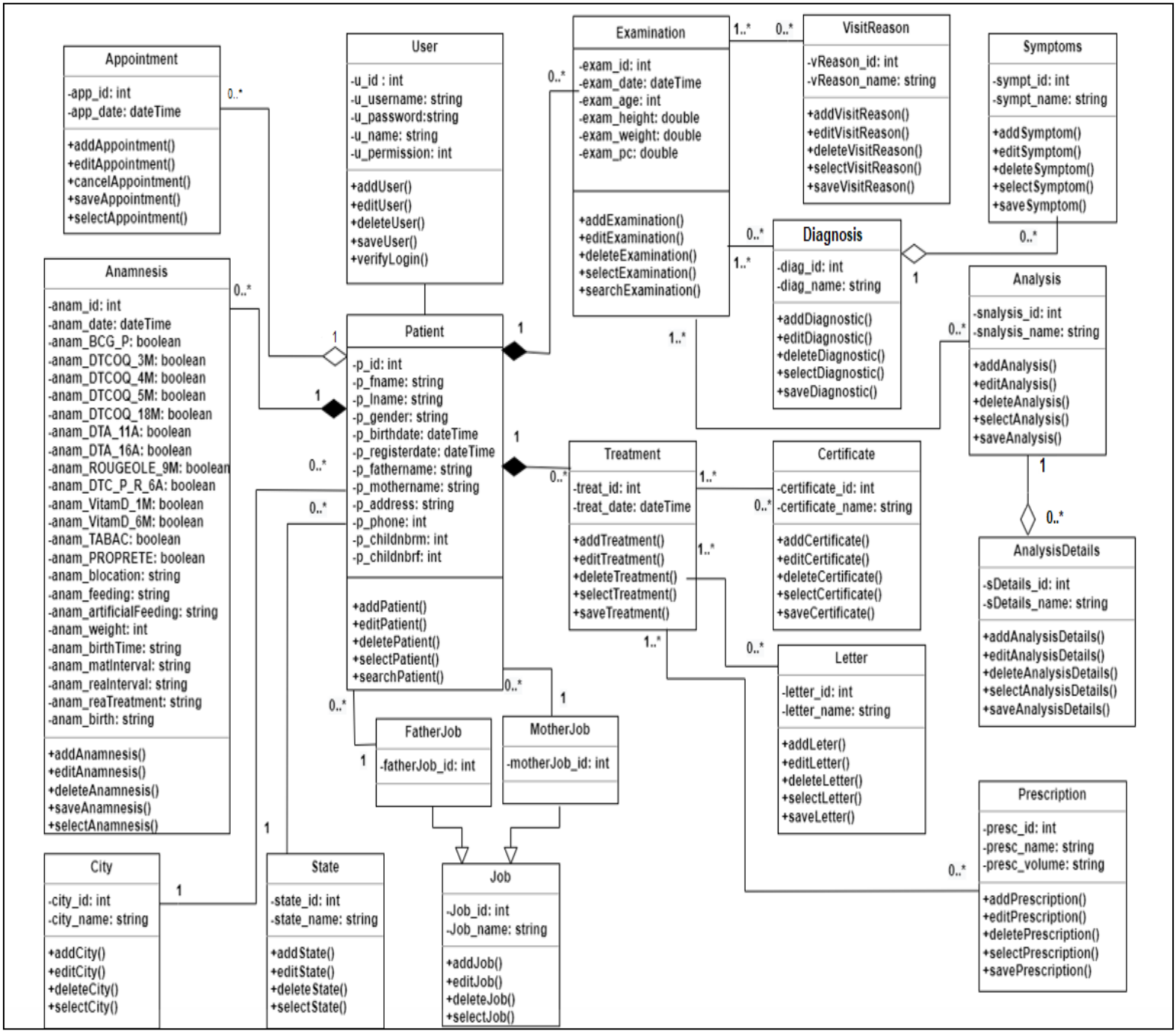

Fig 2. 5 Application's class diagram

2.5 Sequence Diagram

2.5.1 Definition

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. [19]

2.5.2 Sequence diagram notations

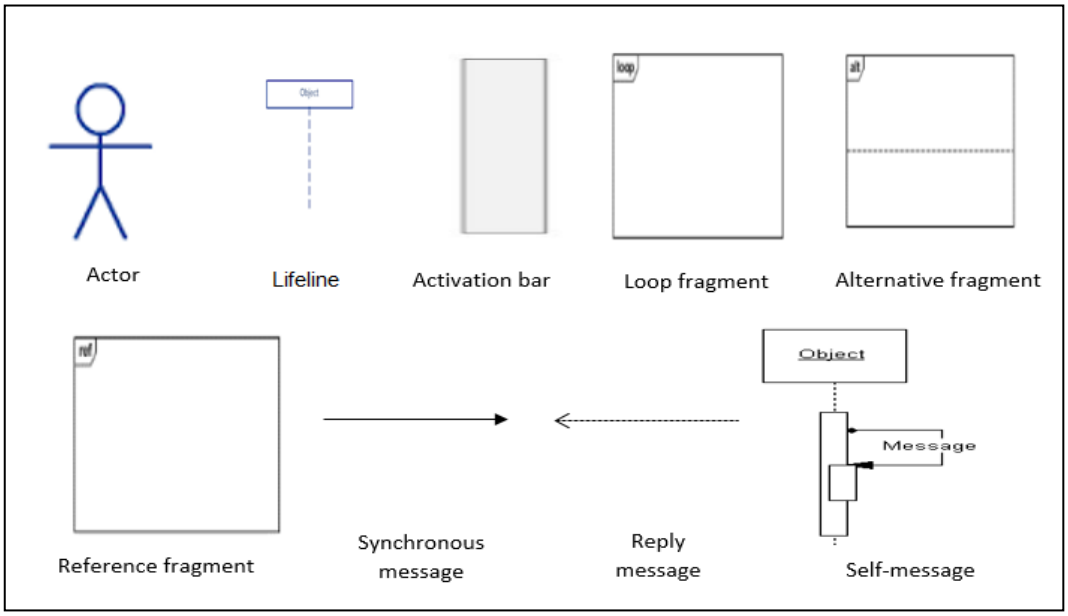

Fig 2. 6 Sequence diagram notations

2.5.2.1 Actors: An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject. Actors may represent roles played by human users, external hardware, or other subjects. [20]

2.5.2.2 Lifelines: A lifeline is a named element which depicts an individual participant in a sequence diagram. Each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram. [19]

2.5.2.3 Activation bar: Activation bar is the box placed on the lifeline. It is used to indicate that an object is active (or instantiated) during an interaction between two objects. The length of the rectangle indicates the duration of the objects staying active. [21]

2.5.2.4 Loop fragment operator: It represents a loop. The loop operand will be repeated a number of times. [20]

2.5.2.5 Alternative fragment operator: It represents a choice of behavior. At most one of the operands will be chosen. [20]

2.5.2.6 Reference fragment operator: It is used to manage the size of large sequence diagrams. It allows reusing a part of one sequence diagram in another. [21]

2.5.2.7 Messages: Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram. Messages can be broadly classified into the following categories: [19]

- **Synchronous messages:** A synchronous message waits for a reply before the interaction can move forward. The sender waits until the receiver has completed the processing of the message. The caller continues only when it knows that the receiver has processed the previous message. [19]
- **Self-message:** Certain scenarios might arise where the object needs to send a message to itself. Such messages are called self-messages. [19]
- **Reply Message:** Reply messages are used to show the message being sent from the receiver to the sender. [19]

2.5.3 Application's sequence diagrams

In this section, sequence diagrams of the most important scenarios that occur in the system are presented:

2.5.3.1. Sequence diagram 01: Authentication

It is the first step after launching the application, if the user enters the correct username and password he will be able to access the application; otherwise, an error message will appear.

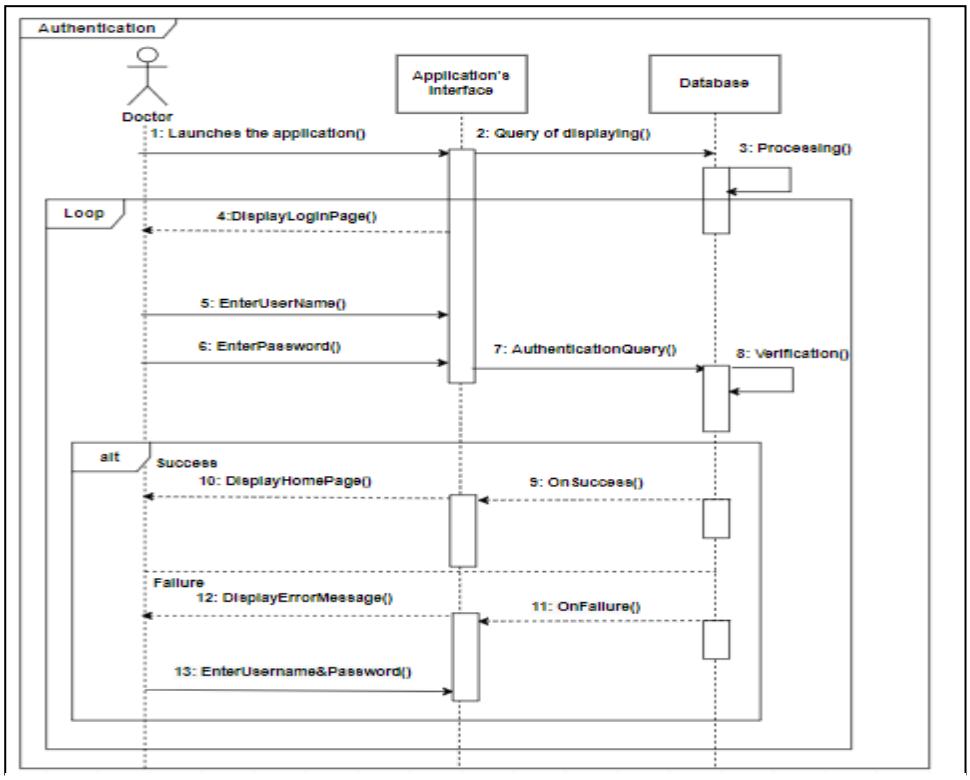

Fig 2. 7 Authentication's sequence diagram

2.5.3.2. Sequence diagram 02: Search page

The Search page give the doctor the possibility to search for an existing patient by name, address or by birthdate. The doctor can also add a new patient by clicking the "Add" button.

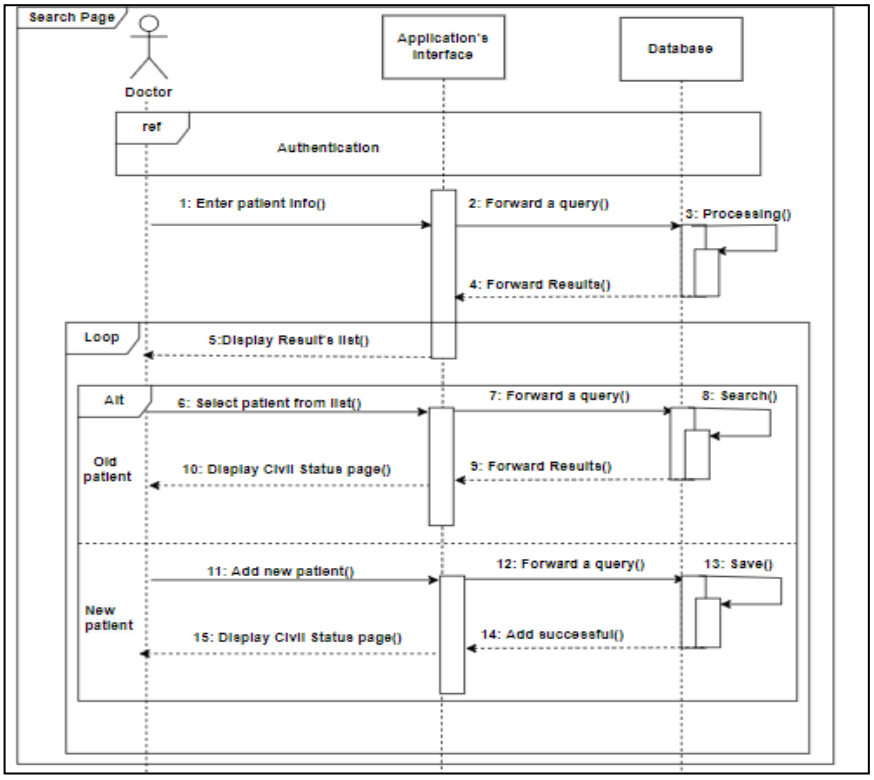

Fig 2. 8 Search page's sequence diagram

2.5.3.3. Sequence diagram 03: Anamnesis page

The Anamnesis page contains medical information about the birth of the patient and whether he is vaccinated or not, the doctor can access this page directly by clicking “Anamnesis” button and entering the patient’s name. If the doctor edits any information, he is required to click “Save” button that is responsible for sending query to database to refresh it and save the changes made.

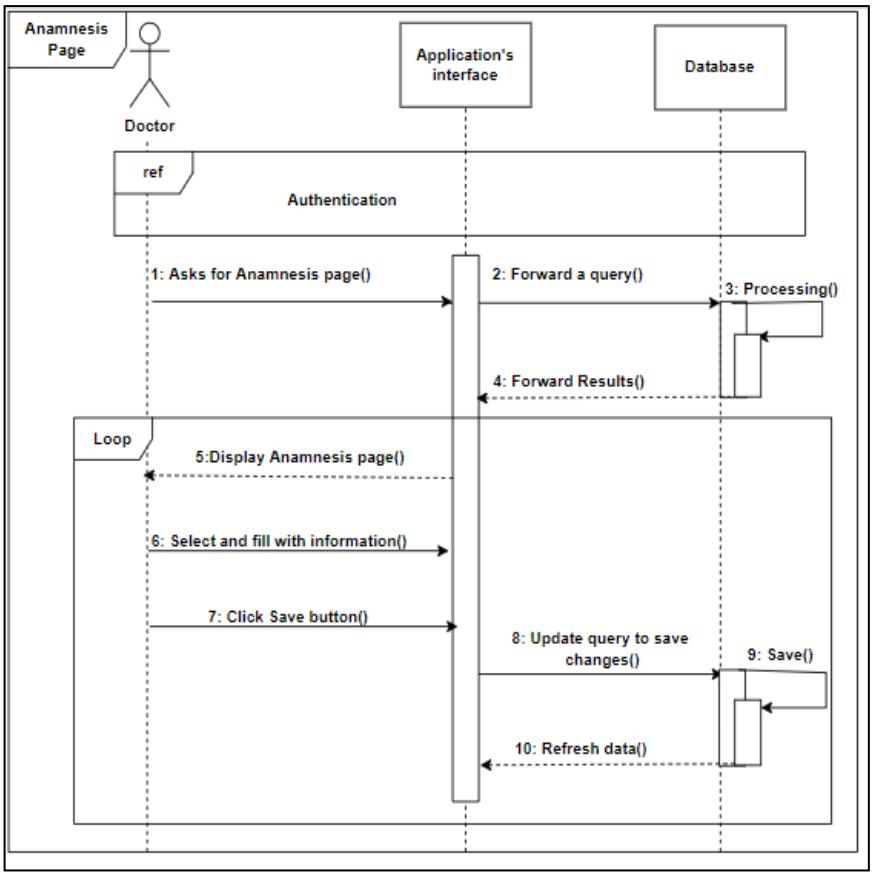

Fig 2. 9 Anamnesis’ sequence diagram

2.5.3.4. Sequence diagram 04: Examination page

When a patient comes to the doctor’s office, the doctor uses the Examination page to fill it with relatable information such as reason of visit, symptoms and diagnosis. These information are either chosen from a given list or added by the doctor and saved so they can be chosen directly the next time.

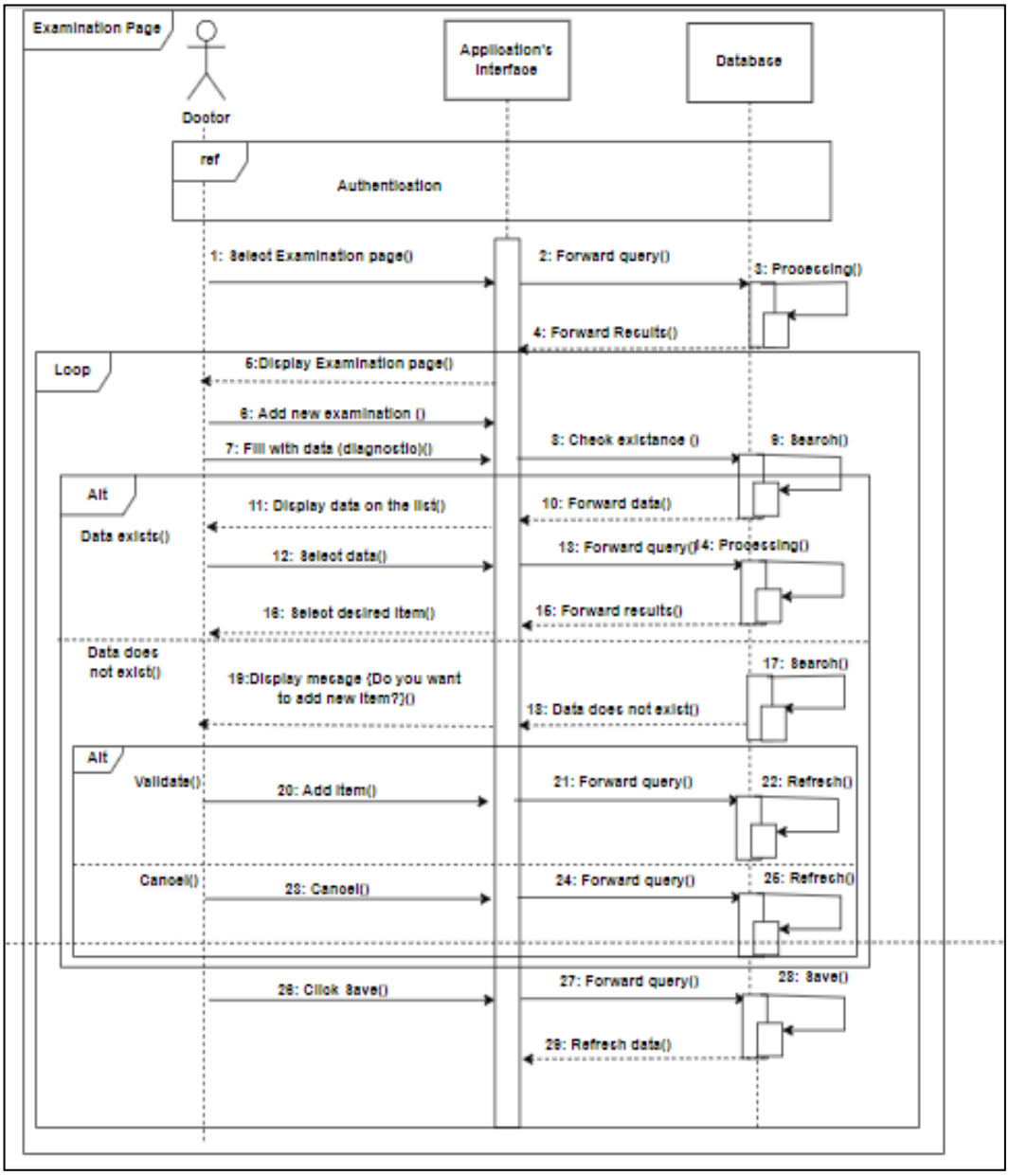

Fig 2. 10 Examination's sequence diagram

2.5.3.5. Sequence diagram 05: Treatment page

After performing the examination, the doctor has to prescribe the appropriate medication to the patient. The treatment is either chosen from a given list or added by the doctor and saved so they can be chosen directly the next time.

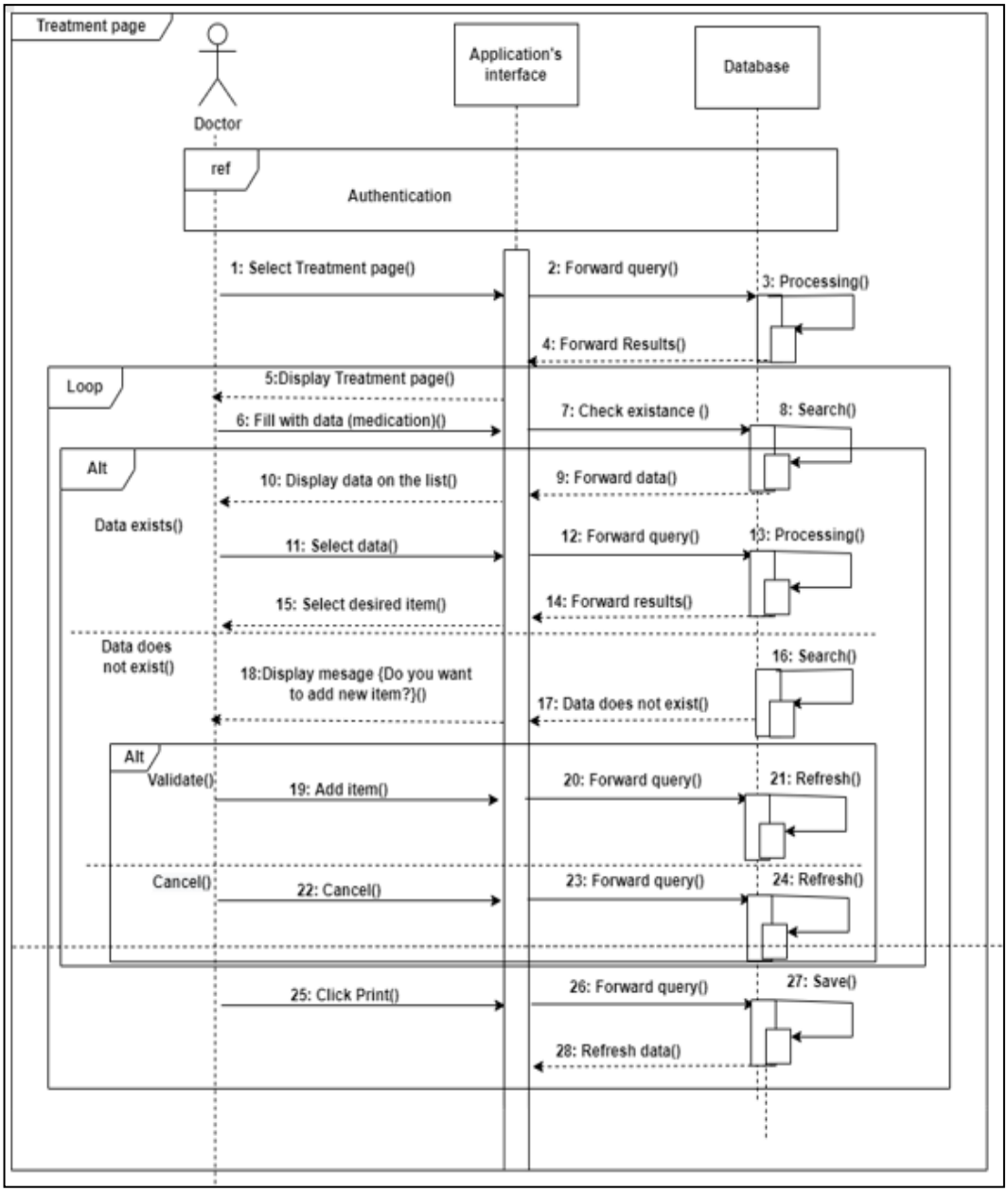

Fig 2. 11 Treatment's sequence diagram

2.5.3.6. Sequence diagram 06: Statistics page

The statistics page offers a number of statistics and graphs related to the application's database. The doctor has the choice to sort the statistics as he wants, for example he can sort patients by sex and display the results in a pie chart.

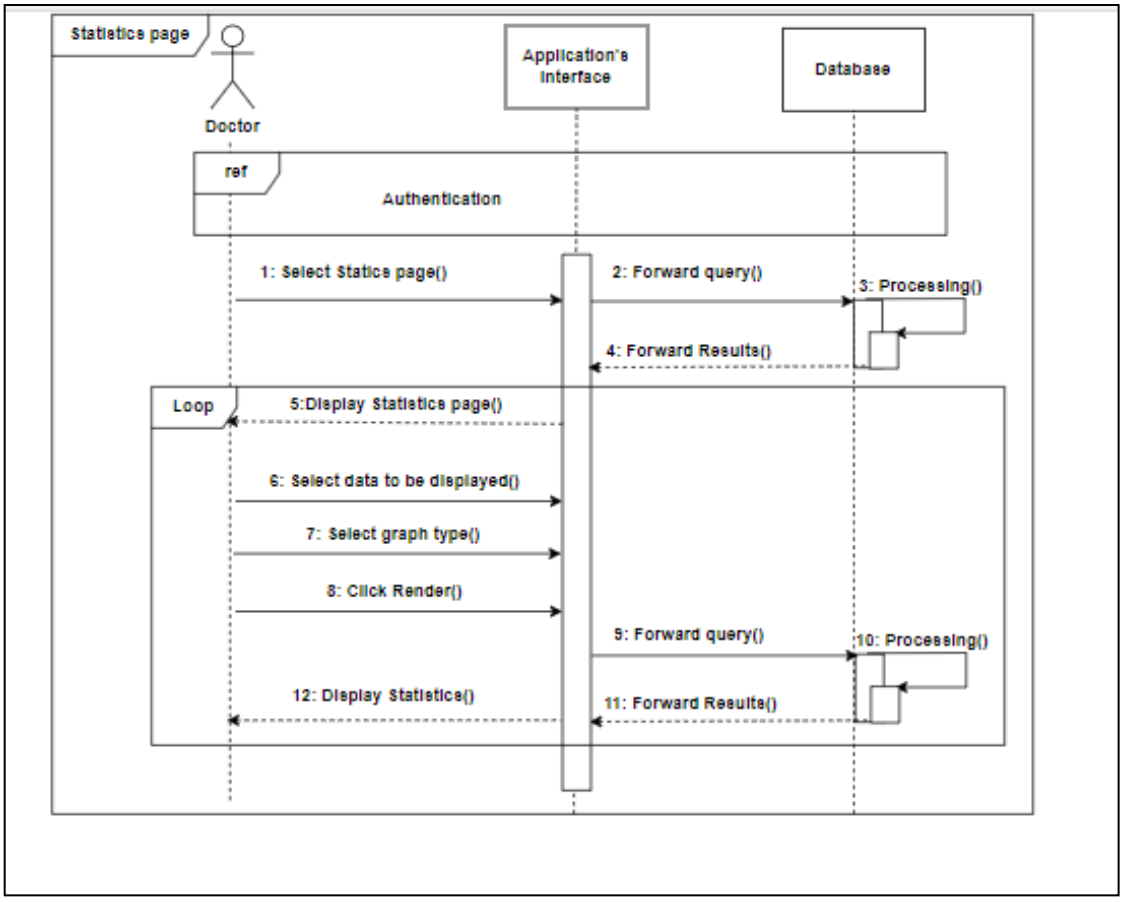

Fig 2. 12 Statistics' sequence diagram

2.5.3.7. Sequence diagram 07: Accounts page

The accounts page's main purpose is to allow the doctor to edit information of an existing account or adding a new account.

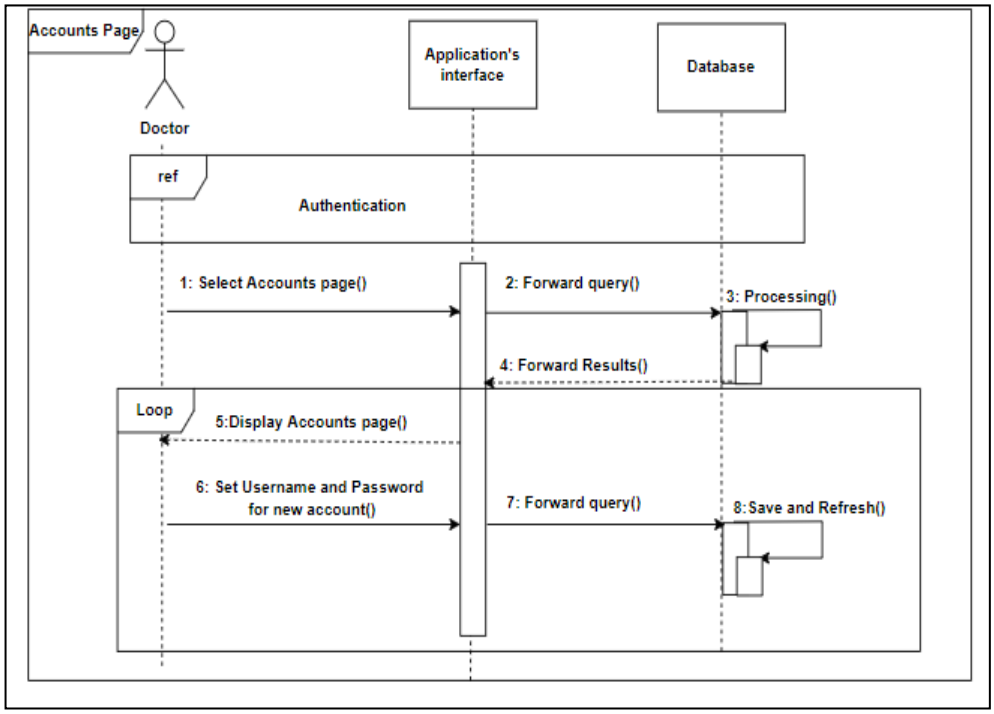

Fig 2. 13 Accounts' sequence diagram

2.6 Introduction to databases

2.6.1 Definition

A database is a collection of information that is organized so that it can be easily accessed, managed and updated. Computer databases typically contain aggregations of data records or files that can be managed by database management system (DBMS). [22]

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications. [23]

2.6.2 Relational model

2.6.2.1 Definition

The relational model represents the database as a collection of relations. Informally, each relation resembles a table of values that consists of rows and columns. [24]

2.6.2.2 Relational model concepts

- **Tables:** relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns.
- **Attribute:** the columns heads in a Table. Attributes are the properties which define a relation.
- **Column:** The column body which represents the set of values for a specific attribute.
- **Tuple:** It is nothing but a single row of a table, which contains a single record.
- **Relation Schema:** A relation schema represents the name of the relation with its attributes.
- **Degree:** The total number of attributes which in the relation.
- **Cardinality:** Total number of rows present in the Table.
- **Relation instance:** Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
- **Relation key:** Every row has one, two or multiple attributes, which is called relation key.

- **Attribute domain:** Every attribute has some pre-defined value and scope which is known as attribute domain. [25]

2.7 Converting class diagram relationships

2.7.1 One-to-Many

The one-to-many relations in our class diagram are as follows:

- Patients - Anamnesis: One patient may have many anamnesis.
- Patients - Examination: One patient may have many examinations.
- Patients - Treatment: One patient may have many treatments.
- FatherJob - Patients: One FatherJob may show up in many patients.
- MotherJob - Patients: One MotherJob may show up in many patients.
- Patients - Appointment: One patient may have many appointments.
- Diagnosis - Symptoms: One disease may have many symptoms.
- Analysis - AnalysisDetails: One analysis may have many analysis details.

To describe how we transferred the class one-to-many relation into relational database tables we take the “Patients - Anamnesis” as an example explained bellow:

To support a one-to-many relationship, we need to design two tables: a table Patient to store information about the patient with p_id as the primary key; and a table Anamnesis to store information about anamnesis related to each patient with anam_id as the primary key. We can then create the one-to-many relationship by storing the primary key of the table Patient (the "one"-end or the parent table) in the table anamnesis (the "many"-end or the child table) as a foreign key. A foreign key of a child table is a primary key of a parent table, used to reference the parent table. [26]

2.7.2 Many-to-Many

The one-to-many relations in our class diagram are as follows:

- User - Patient: Many patients may get treatment from many doctors and many doctors may treat many patients.
- Examination - VisitReason: Many examinations may have the same visit reason and many visit reasons may show up in many examinations.
- Examination - Diagnosis: Many examinations may have the same diagnoses and many diagnoses may show up in many examinations.

- Examination - Analysis: Many examinations may have the same analysis and many analysis may show up in many examinations.
- Treatment - Certificate: Many treatments may have the same certificate and many certificate may show up in many treatments.
- Treatment - Letter: Many treatments may have the same letter and many letters may show up in many treatments.
- Treatment - Prescription: Many treatments may have the same prescription and many prescriptions may show up in many treatments.

To describe how we transferred the many-to-many class relation into relational database tables we take the "Treatment - Certificate" as an example explained bellow:

In our database, a Treatment may contain one or more Certificates and a Certificate can appear in many Treatments. This kind of relationship is known as many-to-many.

We begin with two tables: Treatment and Certificate. The table Treatment contains information about the treatment (treat_id, treat_date) with treat_id as its primary key. The table Certificate contains (certificate_id, certificate_name).

To support many-to-many relationship, we need to create a third table (known as a junction table), say CertificateLine, where each row represents a Certificate of a particular Treatment. For the CertificateLine table, the primary key consists of two columns: certificate_id and treat_id that uniquely identify each row. The columns certificate_id and treat_id in CertificateLine table are used to reference Certificate and Treatment tables. [26]

2.8 Database tables

In this application, the database has played a key part in regulating and managing the information used by the user, as well as assuring the program's flexibility and responsiveness.

The database tables along with their attributes and their meaning are described in the following table:

Table 2. 18 Database tables

Table	Attributes	Meaning
User	u_id	User's identification number, considered as the primary key
	u_username	User's username
	u_password	User's password
	u_permission	The permission given to each user: Doctor(0),

		Receptionist(1)
	u_name	User's name
Patient	p_id	Patient's identification number, considered as the primary key
	u_id	Foreign key
	p_fname	Patient's first name
	p_lname	Patient's last name
	p_gender	Patient's gender
	p_birthdate	Patient's birth date
	p_registerdate	Patient's first visit date
	p_fathername	Patient's father's name
	p_mothername	Patient's mother's name
	state_id	Patient's living state, foreign key
	city_id	Patient's living city, foreign key
	p_address	Patient's address
	p_phone	Patient's parent's phone number
	p_childnbrm	Parent's number of male children
	p_childnbrf	Parent's number of female children
	p_fatherJob	Patient's father's job, foreign key
	p_motherJob	Patient's mother's job, foreign key
Anamnesis	anam_id	Anamenesis' identification number, primary key
	anam_bLocation	Birth location (home, hospital…etc)
	anam_feeding	Whether the feeding is natural, artificial or both
	anam_artificialFeeding	What kind of artificial feeding
	anam_reaInterval	Reanimation interval
	anam_reaTreatment	Reanimation treatment
	anam_date	Anamnesis' date of visit
	(anam_BCG	Whether the patient is vaccinated with these vaccines or not
	anam_DTCOQ_3M	
	anam_DTCOQ_4M	
	anam_DTCOQ_5M	
	anam_DTCOQ_18M	
	anam_DTA_11A	
	anam_DTA_16A	
	anam_ROUGEOLE_9M	
	anam_DTC_P_R_6A	
	anam_VitamD_1M	
	anam_VitamD_6M)	
	anam_TABAC	Whether the parent is a smoker or not
	anam_PROPRETE	Patient cleanliness
	anam_matInterval	Maternal interval
	anam_birth	Birth method (normal, cesarean)
	anam_weight	Patient's weight at birth
	anam_birthTime	Before time, in time or after time
	p_id	Foreign key
Examination	exam_id	Examination's identification number, primary key
	p_id	Foreign key

	exam_date exam_age exam_height exam_pc exam_weight	Examination's date Patient's age at examination Patient's height at examination Cranial perimeter at examination Patient's weight at examination
Treatment	treat_id treat_date p_id	Treatment's identification number, primary key Treatment's date Foreign key
Appointment	app_id p_id app_date	Appointment's identification number, primary key Foreign key Appointment's date
FatherJob	fatherJob_id job_id	Patient's father job, primary key Foreign key
MotherJob	motherJob_id job_id	Patient's mother job, primary key Foreign key
Job	job_id job_name	Job's identification number, primary key Job's name
State	state_id state_name	State's identification number, primary key State's name
City	city_id city_name	City's identification number, primary key City's name
Letter	letter_id letter_name	Letter's identification number, primary key Letter's description
LetterLine	letter_id treat_id	Foreign key Foreign key
Prescription	presc_id presc_name presc_volume	Prescription's identification number, primary key Medicine name Medicine volume
PrescriptionLine	presc_id treat_id	Foreign key Foreign key
Certificate	certificate_id certificate_name	Certificate's identification number, primary key Certificate's description
CertificateLine	certificate_id treat_id	Foreign key Foreign key
Analysis	analysis_id analysis_name	Analysis' identification number, primary key Analysis' name
AnalysisLine	analysis_id exam_id	Foreign key Foreign key
AnalysisDetails	aDetails_id analysis_id aDetails_name	Identification number, primary key Foreign key Specifies which type of analysis
Diagnosis	diag_id diag_name	Diagnosis's identification number, primary key Disease's name
DiagnosisLine	diag_id exam_id	Foreign key Foreign key

Symptoms	sympt_id sympt_name diag_id	Symptom's identification number, primary key Symptom's description Foreign key
VisitReason	vReason_id vRreason_name	Identification number, primary key Reason of visit
VisitReasonLine	exam_id vReason_id	Foreign key Foreign key

2.9 Database schema design

The following figure illustrates our application's database schema and its corresponding one-to-many and many-to-many relationships:

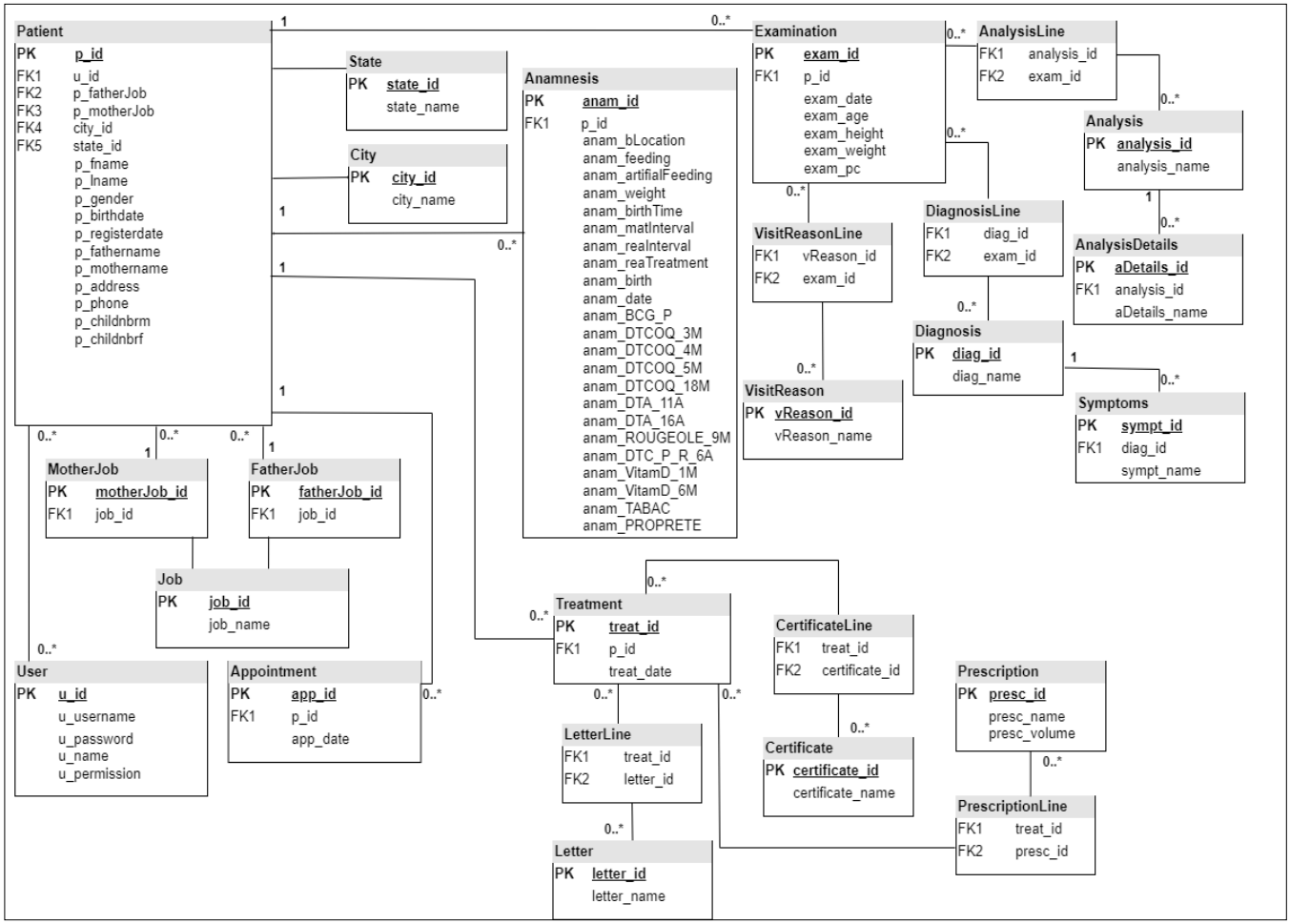

Fig 2. 14 Database schema

2.10 Conclusion

In this chapter, we have presented an overview of UML and its features. We have also shown the UML diagrams we have used in the design of our application, these diagrams include Use-case diagrams, Class diagram and Sequence diagrams. Finally, we have introduced the database and the relational model in addition to converting our class diagram into a database schema.

Chapter Three: System Implementation

3.1. Introduction

In this chapter, we will present the different interfaces of our application along with a detailed description of each interface's functionality. It is to be noted that GUI displaying language is in French as requested by the doctor.

3.2. Interface of "Authentication" page

The "Authentication" page is the door to our application, it is the first form displayed after launching the application and it prompts the user to enter a valid username and password to access the application. If the user enters an invalid username or password an error message will appear and the user should enter them again.

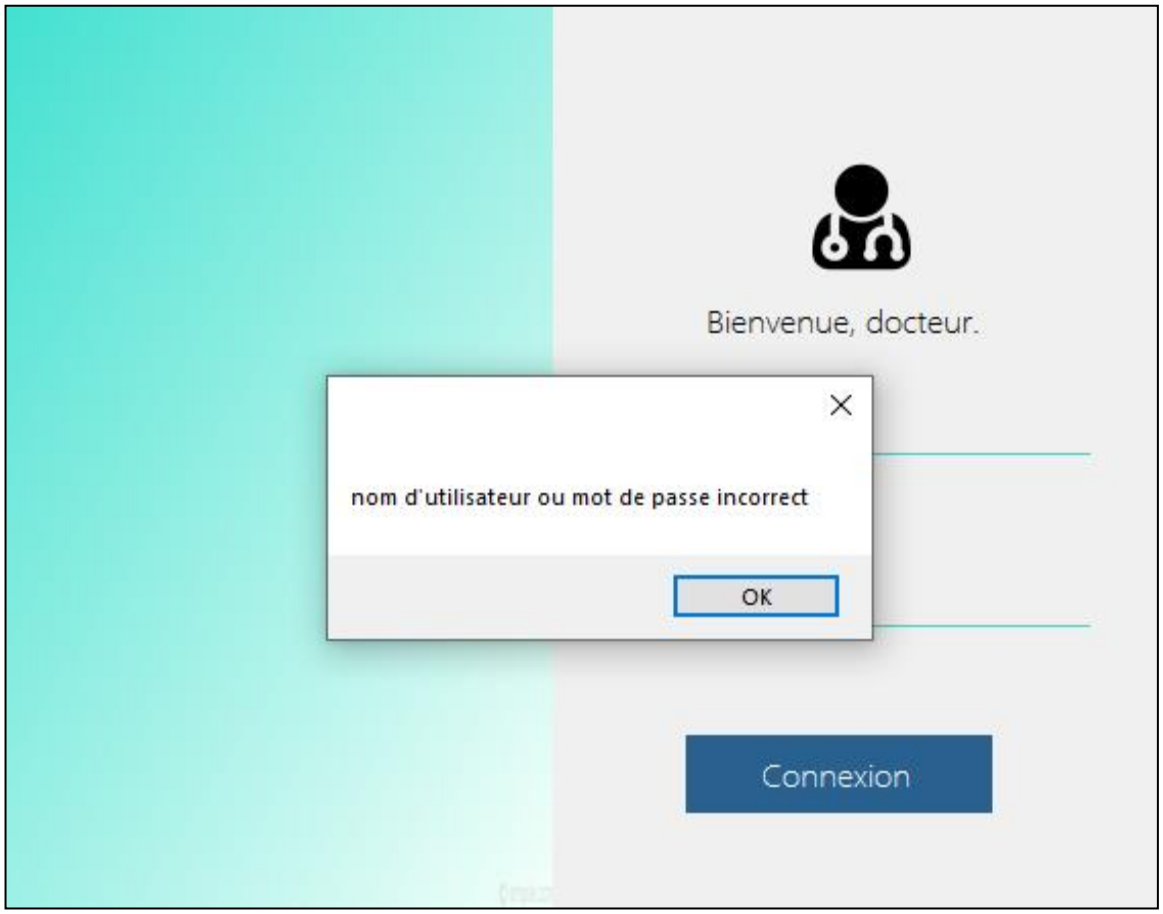

Fig 3. 1 Interface of "Authentication" page

3.3. Interface of the doctor's "Home" page

After the doctor login with his appropriate username and password, the home window of the application will show up along with a side bar containing all the main pages.

The home page displays the number of appointments the doctor has in a specific day, the total number of patients and a list of the doctor's contacts. The list of contacts has been masked for privacy reasons.

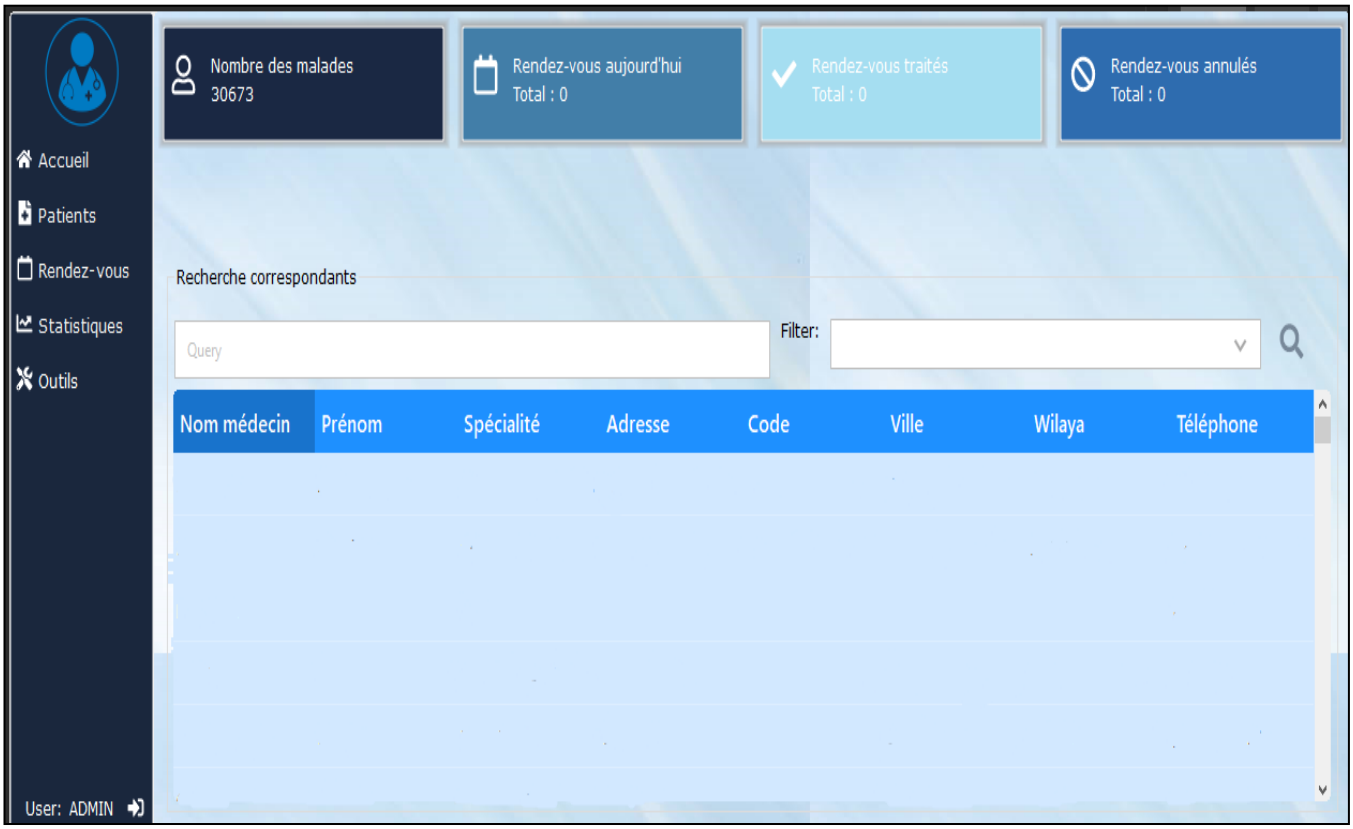

Fig 3. 2 Interface of doctor's "Home" page

3.4. Interface of "Rendez-vous" page

After the receptionist login with his/ her appropriate username, the appointment page will be displayed. This page allows the receptionist to either add a new appointment, edit or cancel an existing one. The "Rendez-vous" page is the receptionist's homepage and the only page he/she has permission to access.

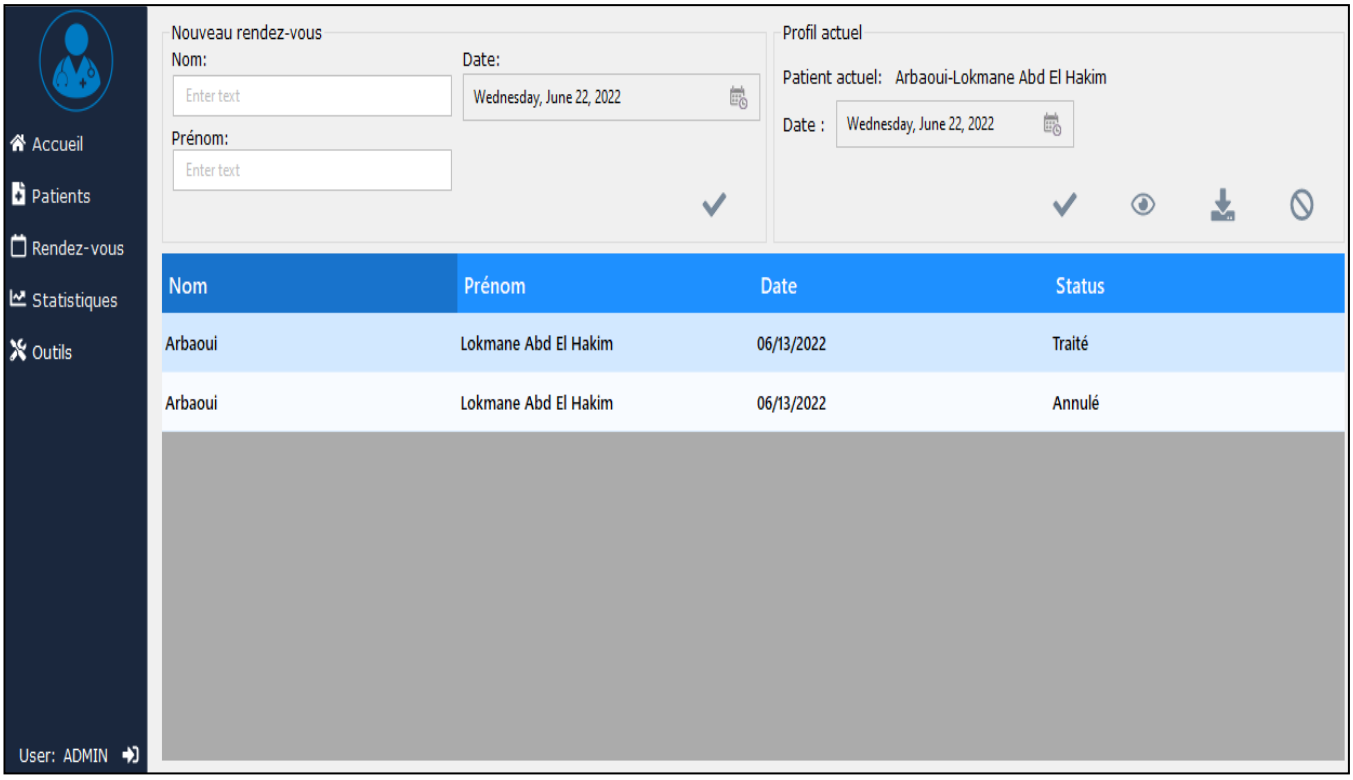

Fig 3. 3 Interface of "Rendez-vous" page

3.5. Interface of "Patients" page

The "Patient" page allows the doctor to perform a search by either name, address or gender; this give the doctor the possibility to filter the patients as needed. After searching, a list of patients will appear and doctor chooses a specific patient.

After selecting the specific patient the doctor either clicks the edit button to edit his information, the delete button to delete the patient's medical record or the view button which will direct the doctor to the medical file of the patient.

If the patient does not exist in the list (a new patient), the doctor clicks the add button which directs him to the "État Civil" page in which the doctor will fill the boxes with the personal information of the patient.

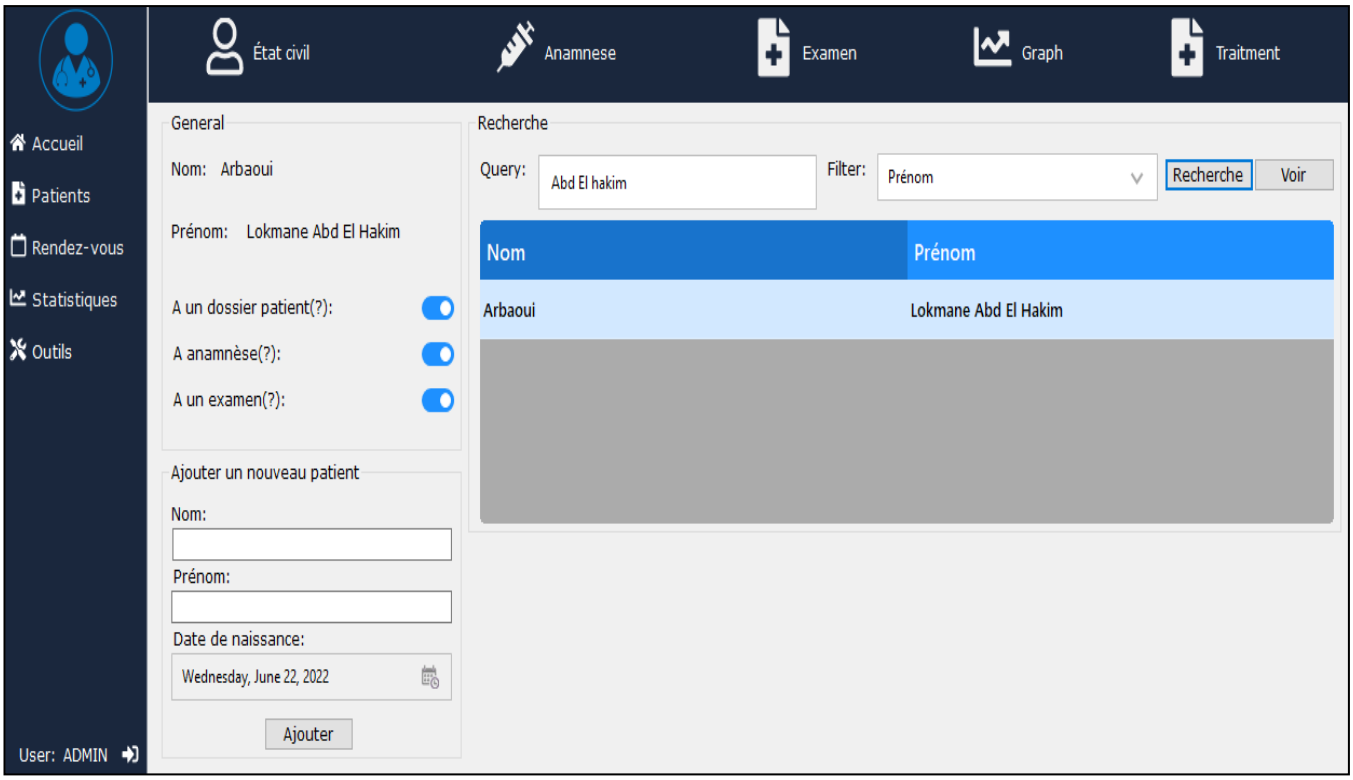

Fig 3. 4 Interface of "Patients" page

3.6. Interface of "État Civil" page

This page appears when the doctor clicks either the add button for a new patient or the view button for an existing patient. It contains information about the patient and his parents. When the doctor finishes filling the patient's information, he clicks the save button to save the changes and jumps to the Anamnesis page.

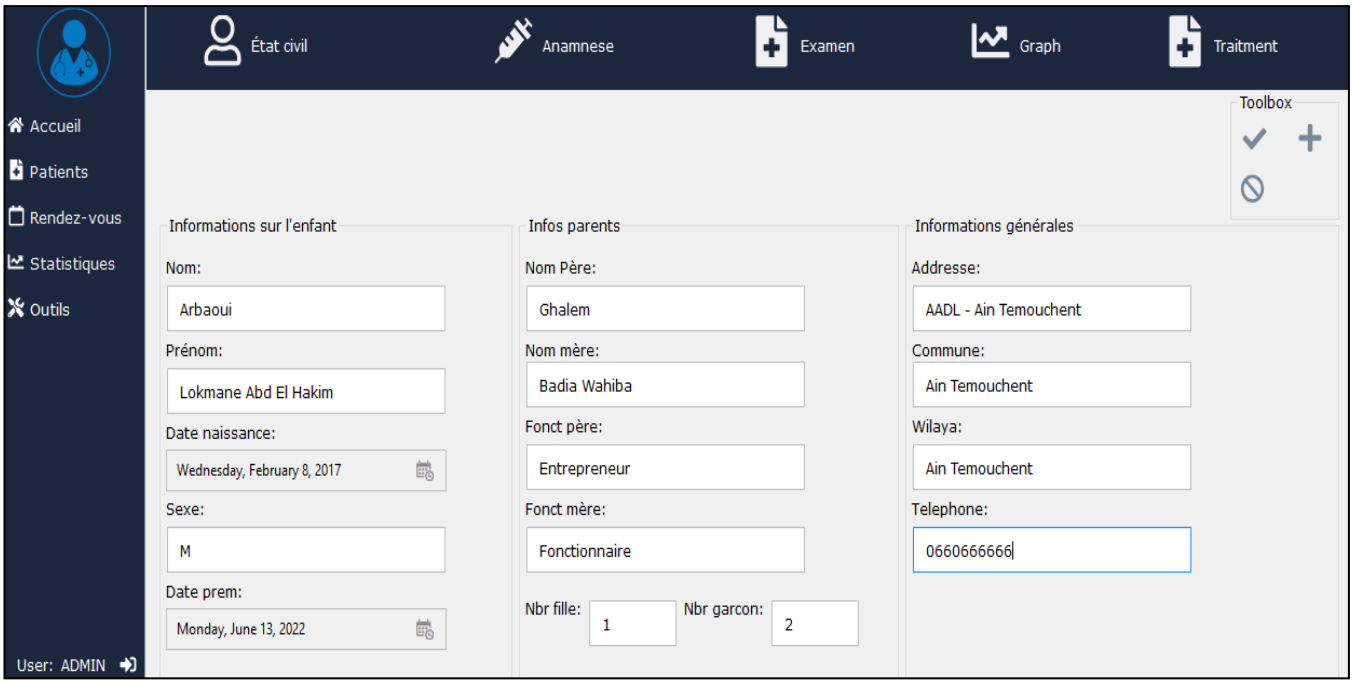

Fig 3. 5 Interface of "État Civil" page

3.7. Interface of "Anamnese" page

The Anamnesis page contains information about the medical background of the patient. After the doctor fills all the related information, he clicks the save button to save the changes.

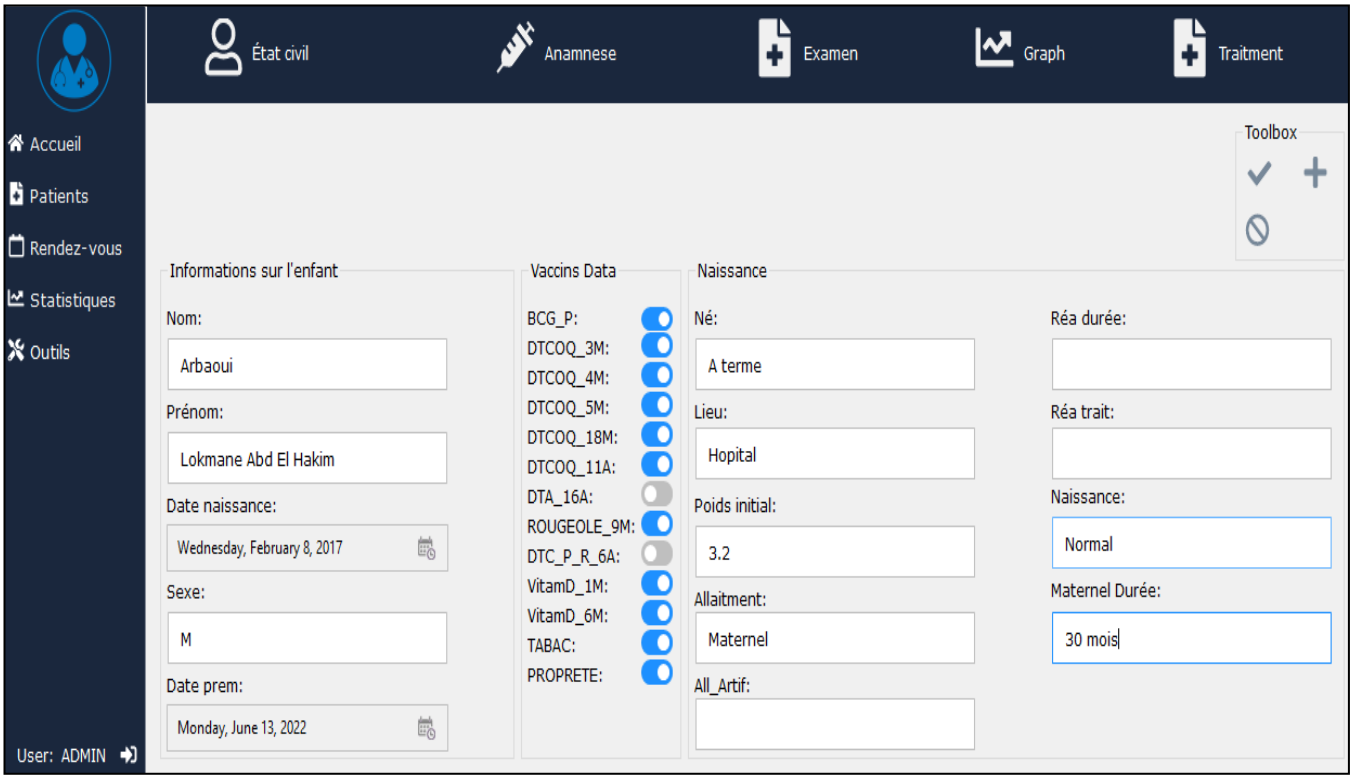

Fig 3.6 Interface of "Anamnese" page

3.8. Interface of “Examen” page

When a patient comes to the doctor’s office, the doctor has to perform a proper examination by analyzing the symptoms and giving a proper diagnosis according to those symptoms. The doctor can also request a number of analysis to be done.

The Examen page contains boxes to fill manually or choose from a given list, starting from the reason of visit, symptoms, diagnosis to analysis and analysis details.

Since the patient may have many examinations, the doctor has to click save each time an examination is done. These examinations are then stored and can be checked later by choosing the corresponding date from a given list.

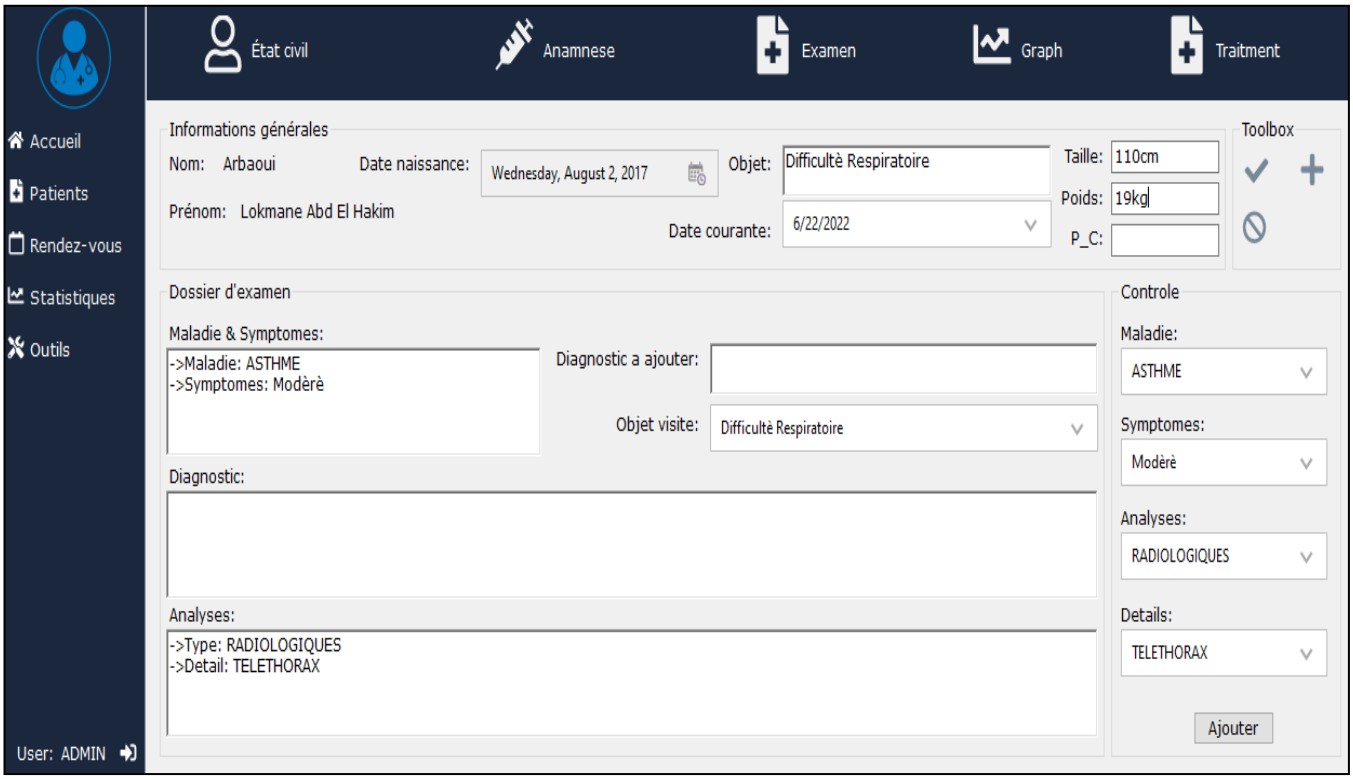

Fig 3. 7 Interface of “Examen” page

3.9. Interface of “Traitement” page

After the examination is done, the doctor moves on to prescribe the medicine to patient. The doctor has to choose the appropriate medicine and volume by either typing them manually or choosing from a given list. Finally, the doctor clicks “Print” button to print the prescription and save the treatment to the medical record of the patient.

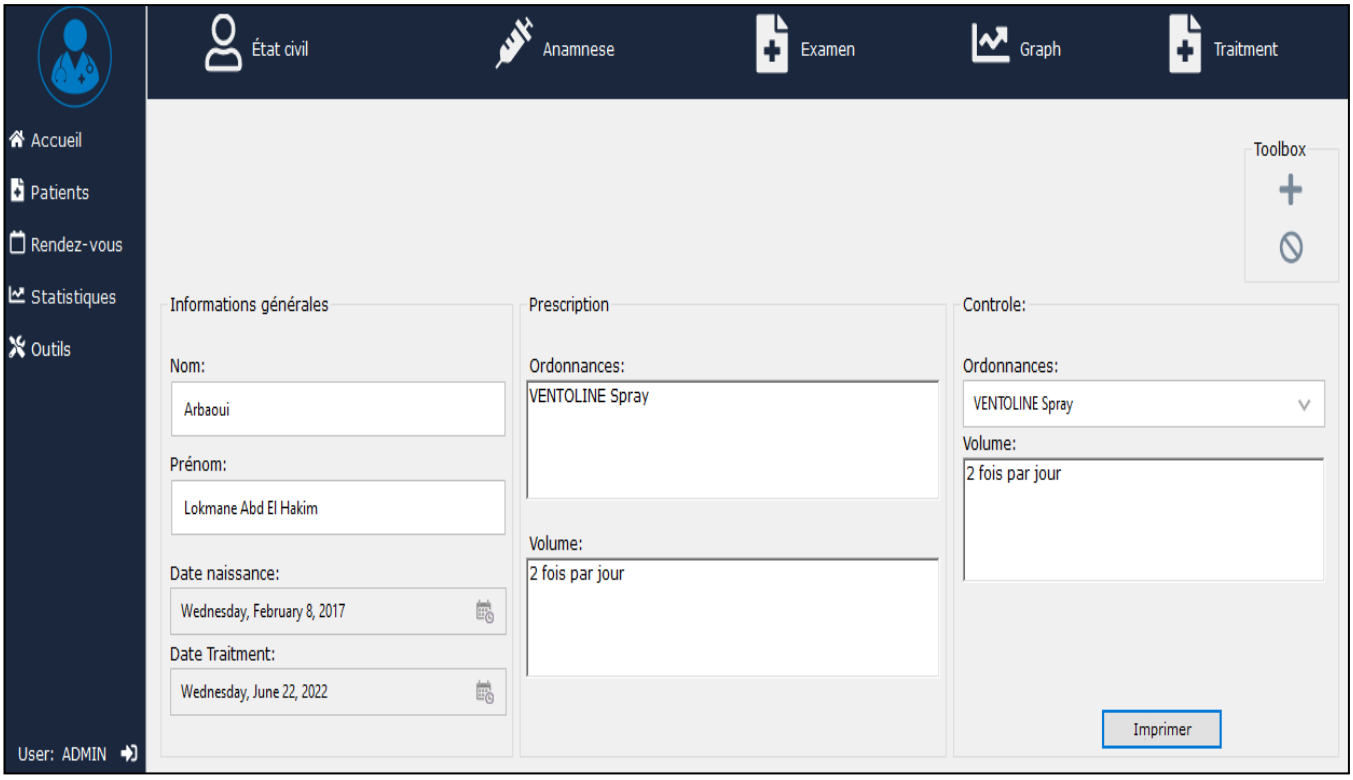

Fig 3. 8 Interface of "Traitement" page

3.10. Interface of "Graph" page

This page provides the doctor with two graphs illustrating the development of the patient's weight and height over time. It also includes two reference figures used to see whether the patient's development is normal or not.

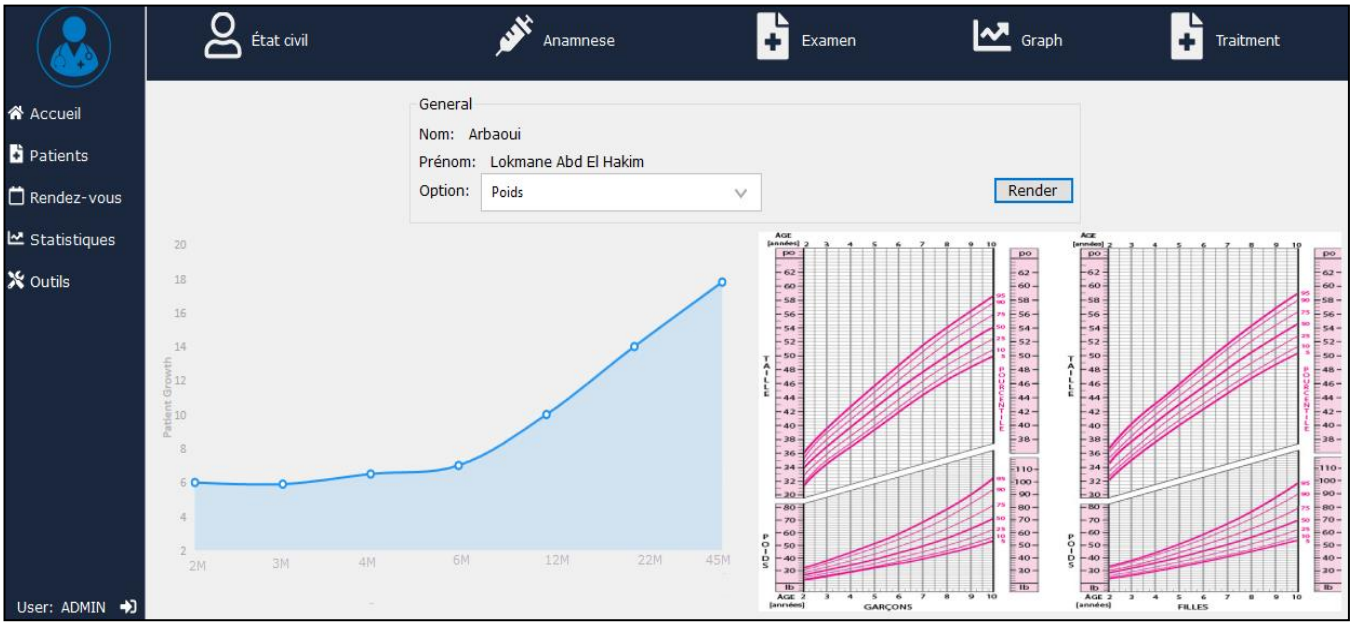

Fig 3. 9 Interface of "Graph" page (weight)

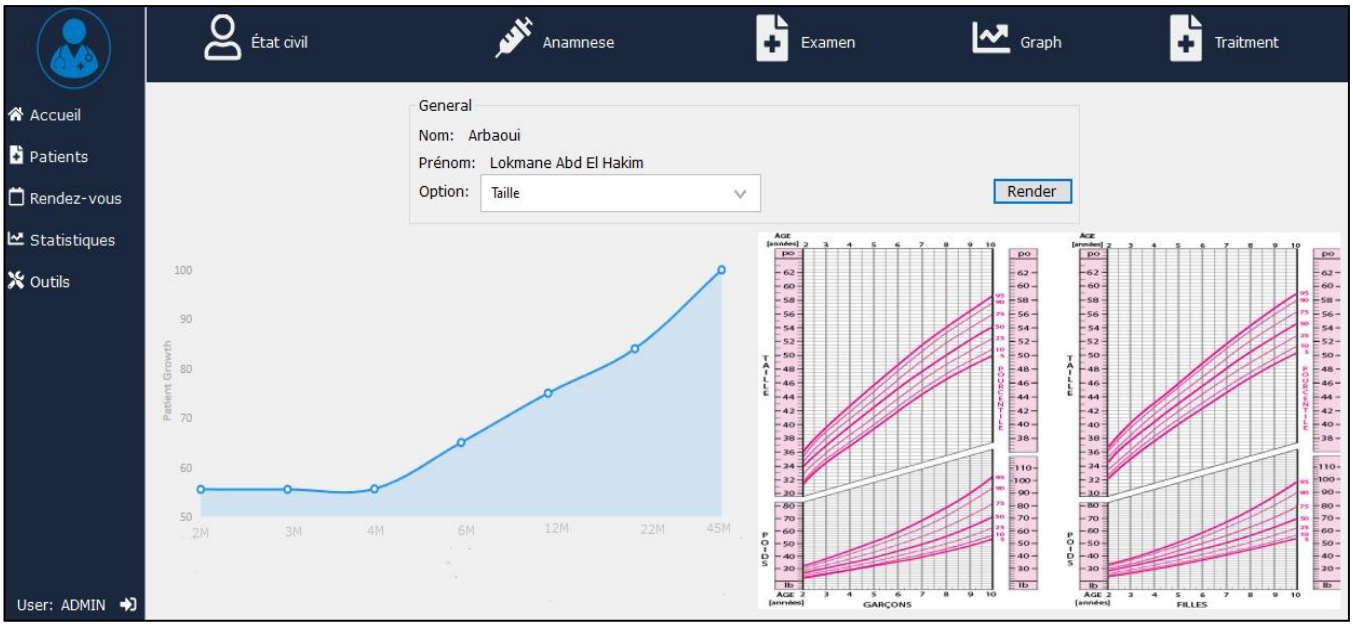

Fig 3. 10 Interface of "Graph" page (height)

3.11. Interface of "Statistiques" page

This page offers a number of statistics related to the patients' database. The doctor can choose among multiple relations given and can specify the graph type in which these statistics will be displayed. The figures below show 2 examples of these statistics.

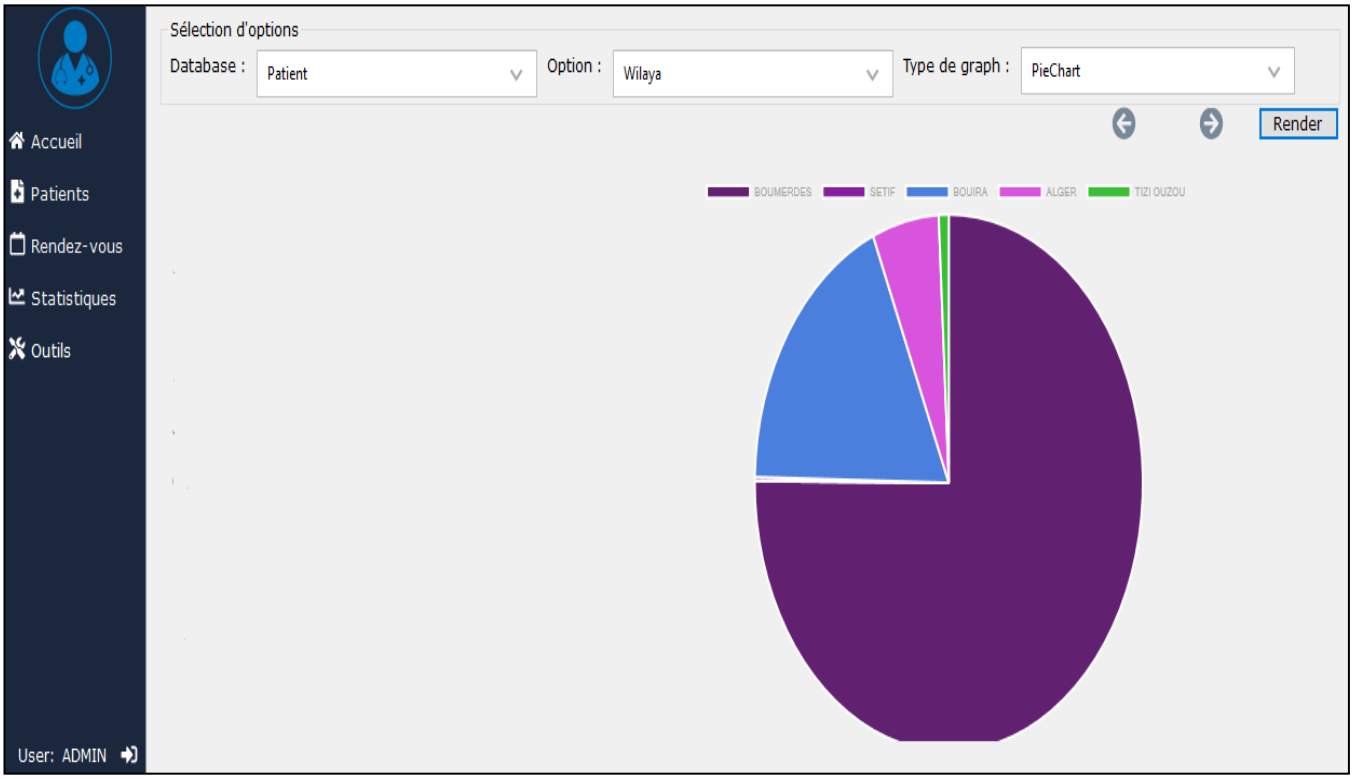

Fig 3. 11 Interface of "Statistiques" page (State)

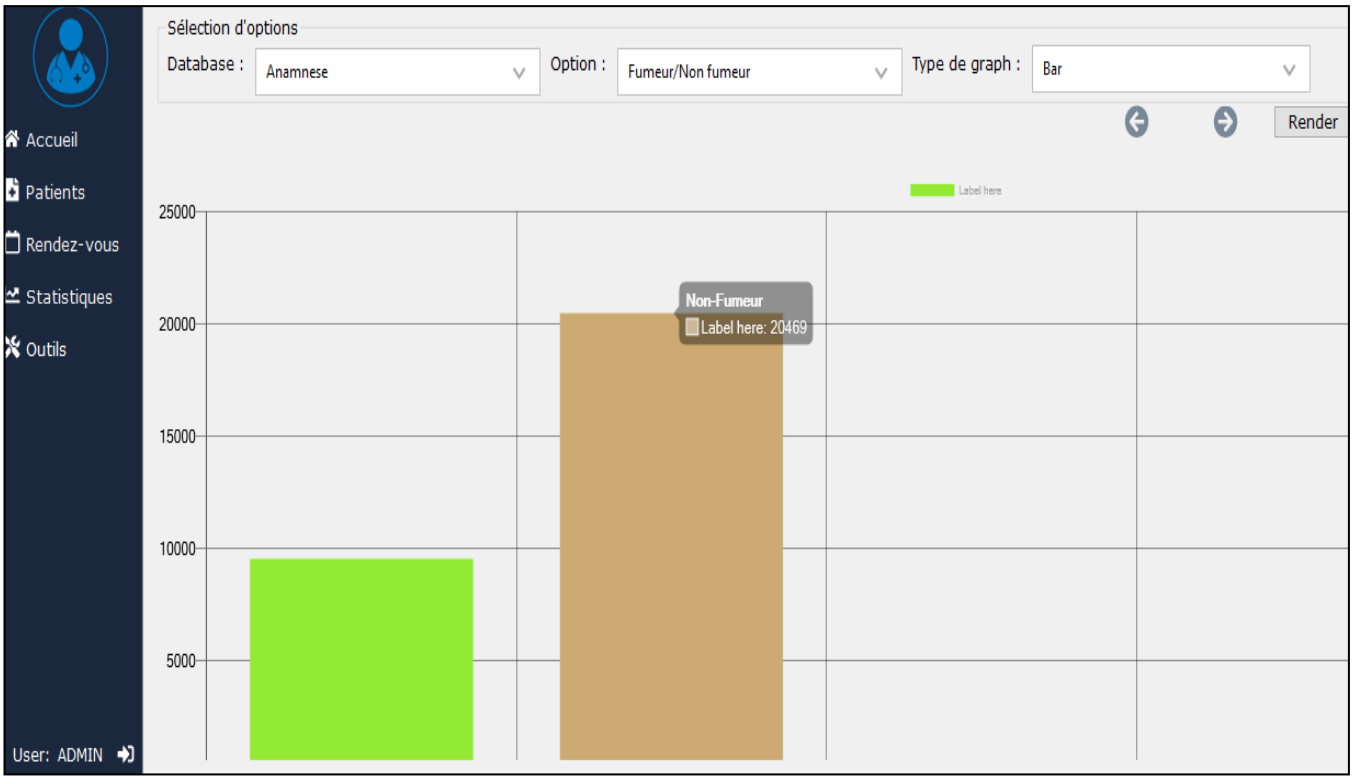

Fig 3. 12 Interface of "Statistiques" page (Smoker/Non-smoker)

3.12. Interface of "Outils" page

The "Outils" page has two main uses: the first one is managing accounts by either adding new accounts or editing information of existing ones. The second use is to add new parameters such as letters, symptoms, diseases…etc

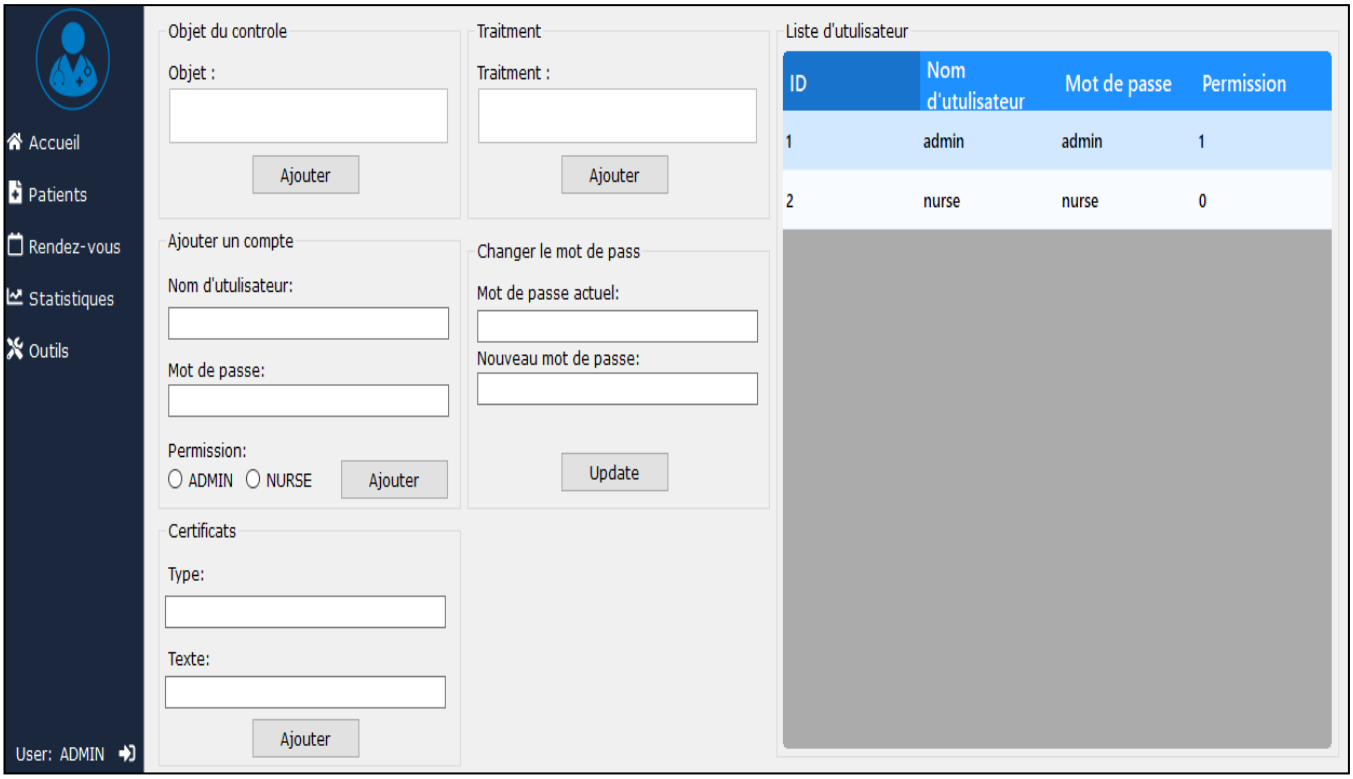

Fig 3. 13 Interface of "Outils" page (State)

3.13. Conclusion

In this chapter, we have presented the multiple interfaces of our application; this includes the functionalities of both the doctor and the receptionist along with a detailed description for each functionality. By this representation, it has been shown that our application is easy to use and user-friendly and that it facilitates the different tasks in the medical office.

General Conclusion

Along with this report, we have illustrated step by step the process of designing and implementing a software management system for a pediatrician's office. After several discussions with the doctor, we have managed to detect the issues he is facing while managing his office and to analyze his requirements in an attempt to implement a satisfactory desktop application that meets his requirements.

The main purpose of this project is to create a desktop application that is secure, reliable, and easy to be used. In addition, this application offers numerous features that effectively facilitate the doctor's job for diagnosis and management.

While working on this project, we have faced many challenges and new concepts that allowed us to enhance our knowledge and skills in the object-oriented programming field.

After achieving the first version of this application, many features are being considered to be added. Some of these improvements include giving permission to the patients to access their medical records, adding a billing system, and more optimization of the application.

Bibliography and Webography

[1] GULATI, H., & SANDHU, H. (1995). A MANAGEMENT INFORMATION SYSTEM FOR MEDICAL SERVICES AT COMMAND HEADQUARTERS.

[2] Ian Griffiths, Programming C# 8.0 (Build Cloud, Web, and Desktop Applications), first edition

[3] https://en.wikipedia.org/wiki/Microsoft_Access#cite_note-ms-import-2. Consulted (15-05-2022)

[4] https://training.nhlearninggroup.com/blog/key-benefits-of-the-microsoft-access-database. Consulted (20/05/2022)

[5] https://www.incredibuild.com/integrations/visual-studio. Consulted (20/05/2022)

[6] https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-6.0. Consulted (21/05/2022)

[7] Anne Boehm and Ged Mead, Murach's ADO.NET 4 database programming with C# 2010

[8] https://docs.microsoft.com/en-us/visualstudio/data-tools/fill-datasets-by-using-tableadapters?view=vs-2022. Consulted (24/05/2022)

[9] https://docs.microsoft.com/en-us/dotnet/desktop/winforms/controls/bindingsource-component-overview?view=netframeworkdesktop-4.8. Consulted (25/05/2022)

[10] Grady Booch, James Rumbaugh, and Ivar Jacobson, The Unified Modeling Language User Guide, 2nd edition

[11] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/behavior-vs-structural-diagram/. Consulted (27/05/2022)

[12] https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case. Consulted (27/05/2022)

[13] https://www.e-education.psu.edu/geog468/l8_p4.html. Consulted (27/05/2022)

[14] https://www.lucidchart.com/pages/uml-use-case-diagram. Consulted (27/05/2022)

[15] Kim Hamilton, Russell Miles, Learning UML 2.0

[16] https://www.uml-diagrams.org/use-case-extend.html. Consulted (27/05/ 2022)

[17] https://www.techopedia.com/definition/16466/class-diagram. Consulted (02/06/2022)

[18] https://www.uml-diagrams.org/visibility.html. Consulted (03/06/2022)

[19] https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/. Consulted (03/06/ 2022)

[20] https://www.visual-paradigm.com/VPGallery/diagrams/Sequence.html. Consulted (04/06/2022)

[21] https://creately.com/blog/diagrams/sequence-diagram-tutorial/. Consulted (04/06/2022)

[22] https://www.techtarget.com/searchdatamanagement/definition/database. Consulted (05/06/2022)

[23] Ramez Elmasri, Shamkant B. Navathe, “Fundamentals of Database Systems”, 6th edition

[24] https://www.relationaldbdesign.com/database-design/module2/relational-database-model.php. Consulted (05/06/2022)

[25] https://www.guru99.com/relational-data-model-dbms.html. Consulted (05/06/2022)

[26] BOUNEZRA Rami, LAKEHAL Mounir, “DESIGN AND IMPLEMENTATION OF A MEDICAL OFFICE MANAGEMENT SYSTEM”, IGEE ex INELEC, University of Boumerdes, Boumerdes, Algeria, 2019.