

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 MEDICAL OFFICE MANAGEMENT
SYSTEM**

Presented by:

- **BOUNEZRA Rami**
- **LAKEHAL Mounir**

Supervisor:

Dr. NAMANE R.

Registration Number:...../2019

Abstract

The Growing number of patients find doctors in the difficulty to manage their individual medical records manually (using paper/card files kept on racks of the cabinet). This way of management leads to a waste of time and effort. In addition, patients' information is unsecure.

The aim of this project is to design and implement a software system (both interactive and friendly-used one) to manage a medical office. Our system helps the doctor to automate the management of his office by providing him/her with all functionalities related mainly to patient management and examination, the electronic storage and rapid access of every patient's individual medical record, treatment prescription, and appointment scheduling.

The medical office of Doctor M.Z. SIDI SAID, a pediatrician practicing in Boumerdes, is considered as case study.

Didications

I dedicate this final project to God Almighty my creator for granting me the wisdom, health and strength to undertake this research task and for enabling me to its completion. To my friends who support me, my brother who encourages me, and to my lovely sister who was here for me in every single moment of my life.

A special feeling of gratitude to my loving parents, my dad who advised me, and helped me to overcome insurmountable obstacles. And of course to my mom who unfortunately could not see this project completed.

Mounir,

I dedicate this work to Almighty God, thank you for your guidance, strength, power of mind, protection and skills, and for giving us a healthy life. To my beloved parents, who have been my source of inspiration and has given me strength when I thought of giving up, who continually provide their moral, emotional, and financial support.

And lastly, to my brothers especially the little one Anass, friends and classmates who shared their words of advice and encouragement to complete my education.

Rami,

Acknowledgements

First and foremost, praises and thanks to God, the Almighty, for His showers of blessings throughout our academic life to complete it successfully.

We would like to express our deep and sincere gratitude to our project supervisor Dr. Namane R. for giving us the opportunity to accomplish this final year project and providing invaluable guidance throughout it. His dynamism, vision, sincerity and motivation have deeply inspired us.

It was a real honor and a great privilege to study in the Institute of Electronics and Electrical Engineering of the University M'Hamed BOUGARA – Boumerdes being one of the best institutes in Algeria .

We are also extremely grateful to our teachers for their support and help to reach this level of knowledge throughout the five years of studies and hard work , especially to those of computer option with whom we learnt about Relational Database Management System (Dr. Khalifat) and Oriented-Object-Programming (Dr. Zitouni) .

Best wishes Thank you .

Table of Contents

Abstract	I
Didications.....	II
Acknowledgements.....	III
Table of Contents	IV
List of Tables	VIII
List of Figures	IX
General Introduction.....	1
Chapter 1: Overview of Pediatrics and Project Objectives	2
1.1 Introduction	2
1.2 Overview of a Pediatrics	2
1.2.1 Definition.....	2
1.2.2 Clinic Organization	2
1.2.3 Staff Role Identification	2
1.2.4 Document Analysis	3
1.3 Problem Statement.....	3
1.4 Objectives.....	4
1.5 Desktop Application Overview	5
1.5.1 Definition.....	5
1.5.2 Advantages and Disadvantages of a Desktop Application.....	5
1.5.3 Why Choosing a Desktop Application	5
1.6 Conclusion.....	5
Chapter 2: Tools, Programming Languages and Technologies Used	6
2.1 Introduction	6
2.2 Development Tools	6
2.2.1 Visual Studio IDE	6

2.2.2 SQL Server Management Studio (SSMS)	6
2.3 Programming language used	6
2.3.1 C# Programming language	6
2.3.2 Structured Query Language “SQL”	7
2.4 Technologies used	7
2.4.1 .NET Framework	7
2.5 Conclusion	10
Chapter 3: System Design	11
3.1 Introduction	11
3.2 Design Requirements.....	11
3.2.1 Modelling language:.....	11
3.2.2 Unified Modelling Language (UML)	11
3.3 Use Case Diagram	13
3.3.1 Definition.....	13
3.3.2 Use Cases	13
3.3.3 Actors	14
3.3.4 Relationships	14
3.3.5 Doctor’s use case diagram design	15
3.3.6 Doctor’s use case diagram textual description	16
3.3.7 Receptionist use case diagram design	19
3.3.8 Receptionist’s use case diagram textual description	20
3.4 Class diagram	21
3.4.1 Definition.....	21
3.4.2 Class Definition	21
3.4.3 Visibility of Class attributes and Operations	21
3.4.4 Class relationship.....	22
3.4.5 Multiplicity	23

3.4.6 Application class list.....	23
3.4.7 Class diagram design.....	28
3.5 Sequence diagram.....	28
3.5.1 Definition.....	28
3.5.2 Notations	29
3.5.3 Sequence diagram 1: Authentication.....	31
3.5.4 Sequence diagram 2: Search page (or add patient)	32
3.5.5 Sequence diagram 3: Anamnesis page	33
3.5.6 Sequence diagram 4: Examination page.....	34
3.5.7 Sequence diagram 5: Treatment page.....	35
3.5.8 Sequence diagram 6: Statistics	36
3.5.9 Sequence diagram 7: Accounts	37
3.6 Introduction to Database.....	38
3.6.1 What is a Relational Database?	38
3.6.2 What is DBMS?.....	39
3.6.3 Database schema	39
3.7 Convert class diagram relationships	39
3.7.1 One-to-Many	39
3.7.2 Many-to-Many.....	40
3.8 Database tables	42
3.9 Database tables meaning	43
3.10 Relational database schema design	47
3.11 Conclusion.....	47
Chapter 4: Implementation	48
4.1 Introduction	48
4.2 Interface of "Authentication" page	48
4.3 Interface of the "Doctor Home page"	49

4.4 Interface of the "Receptionist Home page"	49
4.5 Interface of "Add a patient" page	50
4.6 Interface of "Outils" page.....	51
4.7 Interface of "Comptes" page	51
4.8 Interface of "Etat Civil" page	52
4.9 Interface of "Anamnese" page.....	52
4.10 Interface of "Examen" page:	53
4.11 Interface of "traitements" page.....	54
4.12 Interface of "Graph" page.....	55
4.13 Interface of Statistics page.....	56
4.14 Interface of Receptionist home page	57
4.15 Conclusion.....	57
General Conclusion	58
Webography	59
Bibliography.....	61

List of Tables

Table 1-1 Advantages and disadvantages of the desktop application	5
Table 3-1 UML diagrams type and description.....	12
Table 3-2 Actor and use case relationships	15
Table 3-3 Authentication use case textual description	16
Table 3-4 Patient management use case textual description	16
Table 3-5 Patient examination use case textual description	17
Table 3-6 Treatment use case textual description	18
Table 3-7 Accounts use case textual description.....	18
Table 3-8 Statistics use case textual description	19
Table 3-9 Receptionist authentication use case textual description	20
Table 3-10 Schedule appointment use case textual description	20
Table 3-11 Relationship between two classes	22
Table 3-12 Multiplicity indicator and its meaning	23
Table 3-13 User class description.....	23
Table 3-14 Patient and Anamnesis class description	24
Table 3-15 Examination,Treatment and FatherJob class description.....	25
Table 3-16 MotherJob,Appointment,Analysis class description.....	25
Table 3-17 AnalysisDetails,Diagnostic and DiagSymptom class description	26
Table 3-18 VisitReasonn, Letter and Certificate class description	26
Table 3-19 Prescription, State and City class description	27
Table 3-20 Sequence diagram notations.....	29
Table 3-21 Sequence diagram arrow types	30
Table 3-22 One-to-many relationships	39
Table 3-23 Many-to-many relationship.....	40
Table 3-24 Database table attributes meaning.....	43

List of Figures

Figure 2-1 Interaction of TableAdapter with database	9
Figure 2-2 Demonstration where bindingsource fits into existing data-binding architecture	10
Figure 3-1 Doctor's use case diagram.....	15
Figure 3-2 Receptionist use case diagram	19
Figure 3-3 Application class diagram.....	28
Figure 3-4 Authentication sequence diagram.....	32
Figure 3-5 Search page sequence diagram	33
Figure 3-6 Anamnesis page sequence diagram	34
Figure 3-7 Examination page sequence diagram.....	35
Figure 3-8 Treatment page sequence diagram.....	36
Figure 3-9 Satatistics page sequence diagram.....	37
Figure 3-10 Accounts sequence diagram	38
Figure 3-11 Database schema.....	47
Figure 4-1 Interface of authentication page.....	48
Figure 4-2 Interface of doctor's home page.....	49
Figure 4-3 Interface of receptionist home page.....	50
Figure 4-4 Interface of add patient page.....	50
Figure 4-5 Interface of tools page	51
Figure 4-6 Interface of accounts page	51
Figure 4-7 Interface of civil status page	52
Figure 4-8 Interface of anamnesis page.....	53
Figure 4-9 Interface of examination page	54
Figure 4-10 Interface of treatment page	55
Figure 4-11 Interface of graph page	56
Figure 4-12 Interface of statistics page	56
Figure 4-13 Receptionist home page.....	57

General Introduction

Management Information System is a computer based system that helps managers to manage and organize the organization easily in which it processes information through computers. In order to be successful in your organization, Management Information System is a big help for a better planning and decision. Therefore, this kind of systems is made for a speedy access to accurate data and to help the managers achieve their goals. Furthermore, it creates a great impact on the organization's performance, functions, and of course productivity.

Doctor's office is a private place (family doctor, dermatologist, pediatrician etc.) where medical visits take place for consultation or minor procedures; its staff usually consists of a nurse, receptionist and a doctor.

Currently, many doctors office still store patients' records by using paper or card manual system. Doctors need to manually write down the patients' information and index the patients' medical card, which is kept on the organized racks or in the cabinets. These extra works result in waste of time and manpower. Moreover, patients' information is not secure.

Given this state of affairs, our project aims to design and implement an interactive, reliable, user-friendly desktop application that reduces the burden of the doctors and improve the patient records management system. Our work explores, explains the process in which, an easy to use application software helps to simplify the lives of patient's records management maintained in a database.

The report consists of four chapters. The first chapter introduces the pediatric field's description, problem that may be faced using the existing paper-based system followed by our objectives, in the last point we give an overview of a desktop application. In the second chapter, we give a brief definition of technologies and development tools used in our application. The third one emphasizes the language utilized to design and model our project using many diagrams of the Unified Modeling Language. Finally, the last chapter shows up the result by implementing several interfaces then we end up with a general conclusion.

Chapter 1: Overview of Pediatrics and Project Objectives

1.1 Introduction

This chapter includes a brief description of pediatrics, the organization of its staff and the role of each member; we also specify the problem that this later may face using the existing paper-based system and discussing objectives of our project which aim to ease the work and have a better management system. Finally, we give an overview of a desktop application.

1.2 Overview of a Pediatrics

1.2.1 Definition

Pediatrics is the branch of medicine that involves the medical care of infants, children, and adolescents.

1.2.2 Clinic Organization

There are three main members at a Pediatric office:

- Pediatricians: they are specialized in care and treatment of children ranging from newborns to young adults.
- Receptionist who is employed by the Pediatric clinic to receive visitors and answer telephone calls.
- Patients who are receiving medical care.

1.2.3 Staff Role Identification

The roles of each member are defined as follows:

Receptionist is responsible of:

- Receiving visitors.
- Reminding parents of the appointment scheduled by the doctor.

Pediatricians is responsible of:

- Asking about visit reason.
- Searching for patient existing records.

- Examining patients and determining the diseases.
- Writing a medical prescription.

1.2.4 Document Analysis

The Pediatricians interact with the following documents:

- Medical treatment: means the management and care of a patient to combat disease or disorder by using prescription medications.
- Letters: when pediatrician decides that the patient needs to see another doctor specialized in another domain, in this case he writes a letter that describes the case of the patient.
- Examination request: the pediatrician may need to use result of another examination, which needs materials that do not exist in the clinic. He writes an examination request prescription with some details needed.

1.3 Problem Statement

The paper-based system currently in use causes many problems. When the patient first visits the doctor's office, the doctor is required to fill in a new medical card for this patient including some private information such as name, identity card number, and date of birth, gender, and mailing address.

After the examination process, the doctor needs to write down some diagnosis information and treatments on the medical card. Once again, this medical card is passing to the dispensatory.

After the patient gets their medicine, the doctor will keep his(her) medical card on an organized rack based on index of the card. Usually, these medical cards are arranged in alphabetical order according to the patients' name or based on the reference number for their cards.

The doctor needs to search through the file for the medical card that matches the patient's name for any subsequent visit of the patient. This kind of paper-based system is tedious and tuning. The points below are some problems that have been arisen by using manual system:

- Insecurity: The medical card are just kept on the rack without any security lock thus exposed to unauthorized users who can easily get the vital patient information which are supposed to be confidential .
- Time consuming: the process of searching for medical cards may sometimes cause time wasting also when this last need to be passed from the nurse to doctor and then to dispensatory.
- Space: medical cards need to be stored in the office. When the quantity of cards increases every year, this needs larger storage room accordingly it takes more space.
- Redundant information: sometimes, a patient can have more than one medical card. This happens when the patient forgot whether he /she have been visited, the clinic or not the responsible of registration creates new card.
- Limited capacity: what can be written on the medical card is limited. Doctor cannot include other related information in the card. The card just includes some basic patient information, diagnosis and simple treatment information.
- Treatment sheet handwriting: writing treatment sheets by hand may be incomprehensible for some new pharmacists.

1.4 Objectives

In accordance with the problems mentioned above our project consists of helping to solve these weaknesses, and the main objectives can be summarized as follows:

- Assist doctors in managing patient record.
- Automate the process of manipulating data.
- Generate report (treatment, letters and results) and chart automatically.
- Secure patient's info and protect records.
- Optimize time and space.

1.5 Desktop Application Overview

1.5.1 Definition

An application that runs on a stand alone in a desktop or laptop computer under an operating system generally Microsoft Windows OS. In contrast with "Web-based application", which requires a Web browser to run, and "Mobile application" which runs in smartphones or tablets.

1.5.2 Advantages and Disadvantages of a Desktop Application

Table below lists some advantages and disadvantages of desktop application.

Table 1-1 Advantages and disadvantages of the desktop application

Advantages	Disadvantages
Reduces the time and cost	Run-time errors and bugs because it's the first release which needs updates and upgrades
Safe and secure method and place to store and access data	Difficult to move from one machine to another. Need a knowledge in databases
Easy control operations inserting, deleting, updating	Some of user requirements and functions may not be added due to software limit of developing time or costs

1.5.3 Why Choosing a Desktop Application

The pediatrician works only at his office; he does not need a distance access to the patients' record. Web browsers consume memory without viewing any website. Rather than develop a web-based application that runs over a web browser, we developed a desktop application to save memory and limit data access to the machine holder.

1.6 Conclusion

Through this first chapter, we described the pediatrics office by specifying the role of each member of staff, we also identified the problems that can be faced during the work by giving solutions for them, and finally we gave a brief definition of a desktop application with some advantages.

Chapter 2: Tools, Programming Languages and Technologies Used

2.1 Introduction

This second chapter details the tools needed to successfully build a desktop application in addition to the programming languages and software materials used in order to construct and design our project.

2.2 Development Tools

2.2.1 Visual Studio IDE

The Visual Studio integrated development environment is a creative launching pad that is used to edit, debug, and build code of an application. An integrated development environment (IDE) is a feature-rich program that can be used for many aspects of software development. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to ease the software development process. [1]

2.2.2 SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure, from SQL Server to Azure SQL Database. SSMS provides tools to configure, monitor, and administer instances of SQL Server and databases. SSMS is sufficient to deploy, and upgrade the data-tier components used by applications, as well as build queries and scripts.

Use SSMS to query, design, and manage databases and data warehouses, wherever they are - on your local computer, or in the cloud. . [2]

2.3 Programming language used

2.3.1 C# Programming language

C# is a general-purpose, modern and object-oriented programming language pronounced as “C Sharp”. C# is among the languages for Common Language Infrastructure. C# is a lot similar to Java syntactically and is easy for users who have knowledge of C, C++ or Java. [3]

2.3.2 Structured Query Language “SQL”

SQL (Structured Query Language) is a specialized programming language, which is standardized to be used for managing relational databases and performing various operations on the data. There are various uses of SQL which includes modifying database table and index structures; adding, updating and deleting rows of data; and retrieving various subsets of information from a database for transaction processing and analytics applications. There are specialized queries and operations which operates in the form of commands and commonly known as SQL statements like select, add, insert, update, delete, create, alter and truncate.

SQL became standard programming language for relational databases after they emerged in the late 1970s and early 1980s. Commonly known as SQL databases, relational systems comprise a set of tables which contain data in form of rows and columns, where each column in a table corresponds to a category of data for example, customer name or address & each row contains a data value for the intersecting column.

SQL is also a domain-specific language used for programming and designing in data held in a relational database management system (RDBMS). It is particularly useful in handling structured data where there are relations between different entities/variables of the data. [4]

2.4 Technologies used

2.4.1 .NET Framework

The .NET Framework is a development platform for building apps for web, Windows, Windows Phone, Windows Server, and Microsoft Azure. It consists of the common language runtime (CLR) and the .NET Framework class library, which includes a broad range of functionality and support for many industry standards.

The .NET Framework provides many services, including memory management, type and memory safety, security, networking, and application deployment. It provides easy-to-use data structures and APIs that abstract the lower-level Windows operating system. You can use different programming languages with the .NET Framework, including C#, F#, and Visual Basic. [5]

Over and above namespaces are heavily used in C# programming in two ways.

First, the .NET Framework uses namespaces to organize its several classes.

Second, declaring your own namespaces can help you control the scope of class and method names in larger programming projects.

One of the main namespaces that we have used in our work is “System.Data”. The System.Data namespace provides access to classes that represent the ADO.NET architecture. ADO.NET lets you build components that efficiently manage data from multiple data sources, it contains very interesting classes that we have employed in order to manage the database.

The list below shows a brief description of the components utilized in our project:

Dataset: it is a complex object. Approximately equivalent to an in-memory representation of a database. It contains DataTables that correlate to database tables. These in turn contain a series of DataColumnns that define the composition of each DataRow. The DataRow correlates to a row in a database table. You can also establish relationships between DataTables within the DataSet in the same way that a database has relationships between tables. One of the ongoing challenges for the object-oriented programming paradigm is that it does not align smoothly with the relational database model. The DataSet object goes a long way toward bridging this gap because it can be used to represent and work with relational data in an object-oriented fashion. [6]

TableAdapter: it is a component that fills a dataset with data from the database, based on one or more queries or stored procedures we specify. TableAdapters can also perform adds, updates, and deletes on the database to persist changes that we make to the dataset. TableAdapters are designer-generated components that connect to a database, run queries or stored procedures, and fill their DataTable with the returned data. It also send updated data from your application back to the database. [6]

Figure below shows how TableAdapters interact with databases and other objects in memory.

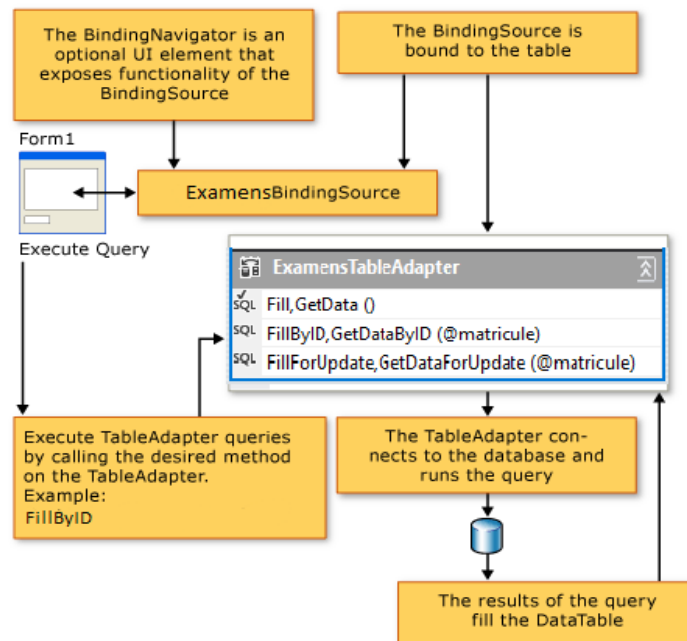


Figure 2-1 Interaction of TableAdapter with database

Binding Data: The most common type of application is one that retrieves data from a database, displays the data, allows changes to be made, and then persists those changes back to the database. The middle steps that connect the in-memory data with the visual elements are referred to as **DataBinding**. [6]

BindingSource: it's a component acts as the data source for some or all of the controls on the form.

You can bind the BindingSource component to both simple data sources, like a single property of an object or a basic collection like ArrayList, and complex data sources, like a database table. The BindingSource component acts as an intermediary that provides binding and currency management services. At design time or run time, you can bind a BindingSource component to a complex data source by setting its DataSource and DataMember properties to the database and table, respectively. [7]

Figure below demonstrates where the BindingSource component fits into the existing data-binding architecture.

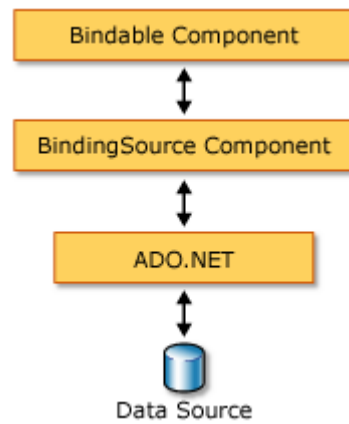


Figure 2-2 Demonstration where bindingsource fits into existing data-binding architecture

2.5 Conclusion

In this chapter, we have mentioned the technologies needed to build our simple project, we also discussed the tools and languages used by specifying some components such as classes and TableAdapter and binding component.

Chapter 3: System Design

3.1 Introduction

System Design is the next step of our project and can only take place after the requirements of the application have been gathered accurately based on several discussions made with the Doctor. Accordingly, we are going to define the role of each actor that interacts with the system. In addition, we will use Unified Modelling Language (UML) for modeling and in particular we choose three diagrams to model these roles: use case, and class diagrams and finally sequence diagram.

3.2 Design Requirements

3.2.1 Modelling language:

A modelling language is mainly used in the field of computer science and engineering for designing models of new software, systems, devices and equipment. UML is a popular modelling language that is used to build system and object models.

The UML gives us the ability to model, in a single language, the application database business and architecture of the system. [8]

3.2.2 Unified Modelling Language (UML)

Unified Modeling language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. UML is an important aspect involved in object-oriented software development. It uses graphic notations to create visual models of software systems. [9]

In addition, it supports high-level development concepts such as frameworks, patterns and collaborations. UML includes a collection of elements such as:

- Programming Language Statements.
- Actors: specify a role played by a user or any other system interacting with the subject.
- Activities: These are tasks, which must take place in order to fulfill an operation contract.

- Logical and Reusable Software Components

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

- Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts. This includes class, composite structure, component, deployment, object, and package diagrams. [10]
- Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time. This includes activity, Sequence, Use case, State, Communication.

In our case we have used three diagrams for modelling, which are defined in Table 3-1 below:

Table 3-1 UML diagrams type and description

Diagram	description	Type
Use case Diagram	Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities. They also provide a good high level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented. [11]	Behavioral

Sequence Diagram	A sequence diagram illustrates how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed. [12]	Behavioral
Class Diagram	Class diagram describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. [13]	Structural

3.3 Use Case Diagram

3.3.1 Definition

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. Actors are people or entities operating under defined roles within the system. [11]

Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation. [14]

3.3.2 Use Cases

Use cases are represented using ovals labeled with verbs that describes the system's functions. In our application, we define the following use cases:

- A. Authentication: Login before accessing the application. Moreover, authentication insures the identity of the user.
- B. Patient Management: Give the possibility to add new patient or edit patient records in anamnesis page.
- C. Patient Examination: Allows the doctor to select (Reason of Visit, Diagnostic, Symptom, Complementary Examination) from the list shown and add new items if needed.

- D. Treatment: Allows a doctor to select (Medicine, Quantity & Dosage) from the list shown and add new items if they do not exist and it permits also to print the prescription after finishing.
- E. Statistics: allows the doctor to get some directory statistics as needed.
- F. Give Permission: allows the doctor to let another user to get access to the application with limited permissions using appropriate username and password.
- G. Scheduling appointments: where the receptionist get minimum patient information in order to call and remind him of the appointment or either change the date.

3.3.3 Actors



Actors are usually individuals involved with the system defined according to their roles. The actor can be a human or other external system. Actors interacting with our application are:

- I. Doctor:
 - Authentication.
 - Patient Management.
 - Patient Examination.
 - Treatment prescription.
 - Accounts creation.
 - Access to statistics.
- II. Receptionist:
 - Scheduling appointments.

3.3.4 Relationships

Relationships between an actor and a use case are illustrated with a simple line. Whereas relationship between two use cases is basically modeling the dependency between the two use cases.

Table 3-2 Actor and use case relationships

Symbols	Description
	"uses" relationship indicates that one use case is needed by another in order to perform a task.
	An "extends" relationship indicates alternative options under a certain use case.

3.3.5 Doctor's use case diagram design

The relationship between the doctor and use cases is shown in the doctor's use diagram in the following figure:

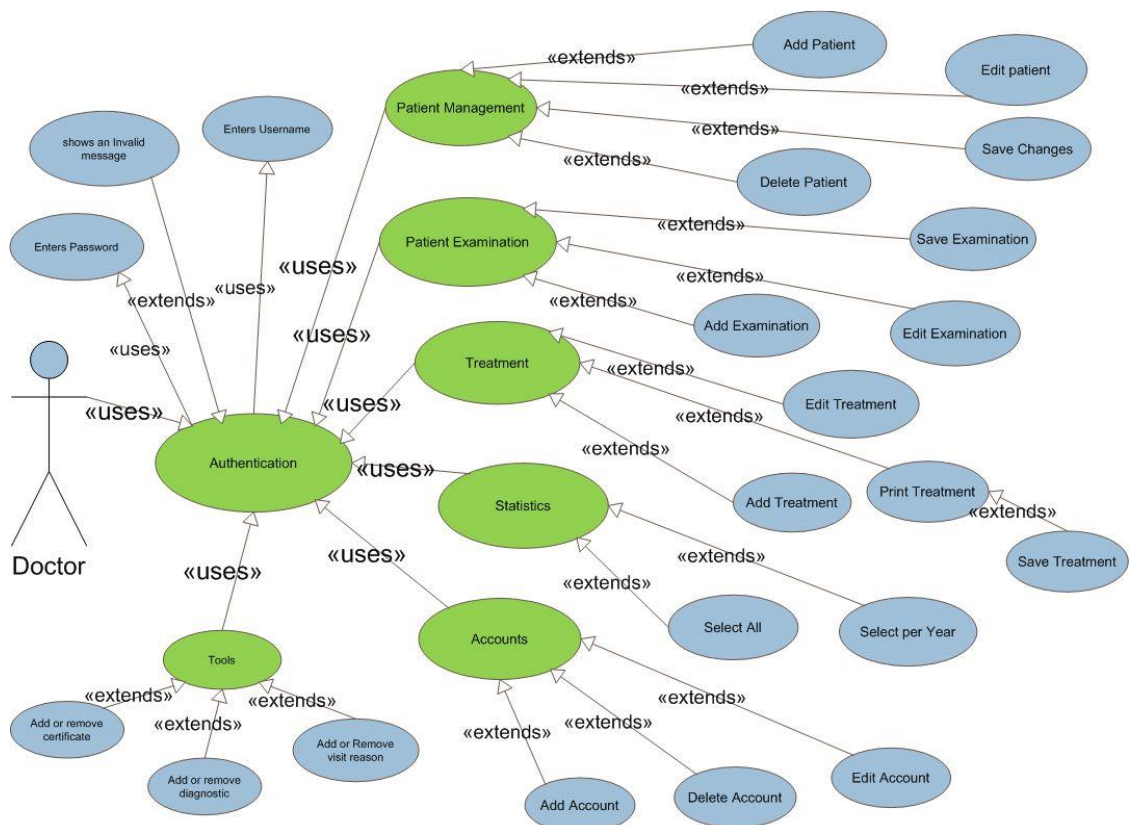


Figure 3-1 Doctor's use case diagram

3.3.6 Doctor's use case diagram textual description

The tables below summarize different use cases that the doctor needs to perform using the application

3.3.6.1 Authentication Use Case

Table 3-3 Authentication use case textual description

Use Case Name	Authentication
Actor	Doctor
Objective	Login and have access to the application
Precondition	Connect to SQL server
Scenario	<ol style="list-style-type: none"> 1.The doctor lunches the application 2.The System prompts the doctor for user name and password 3.The doctor enters his username and password 4. The system checks the conformity of the information entered by sending an authentication query to the local server 5. The server verifies the query and sends favorable answer. 6. The user accesses the application.
Alternative	<p>If the username or password is wrong or missed the system displays "Invalid authentication"</p> <p>message (return to 2)</p>

3.3.6.2 Patient management Use Case

Table 3-4 Patient management use case textual description

Use Case Name	Patient Management
Actor	Doctor
Objective	Add new patient, delete or edit an old patient
Precondition	Authentication

Scenario	<ol style="list-style-type: none"> 1.The doctor selects search page 2.The doctor enters patient name 3. The system checks the existence of the patient in the previous list 4. The doctor selects patient from the list. 5. Civil status page of that patient appears 6.The doctor fill boxes with information related to the patient 7.The doctor jumps to Anamnesis interface 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 in the list, the doctor him directly from the Add button. (goes directly to 5)

3.3.6.3 Patient examination Use Case

Table 3-5 Patient examination use case textual description

Use Case Name	Patient Examination
Actor	Doctor
Objective	Identify the illness by either editing previous or adding new examination page, then save changes
Precondition	Patient Management
Scenario	<ol style="list-style-type: none"> 1.The doctor selects or adds new Examination page 2.The doctor selects a Reason Of Visit from a stored list 3.The doctor selects a 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 asks if it needs to be added so as he can add it

3.3.6.4 Treatment Use Case

Table 3-6 Treatment use case textual description

Use Case Name	Treatment
Actor	Doctor
Objective	Prescribe medicine to patient, and finally print prescription
Precondition	Patient Examination
Scenario	<ol style="list-style-type: none"> 1.The doctor selects Treatment page 2.The doctor selects a Medicine from a stored list 3.The doctor selects a Quantity&Dosage from a stored list 4.The doctor clicks Print button to print prescription and 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 asks if it needs to be added so as he can add it

3.3.6.5 Accounts Use Case

Table 3-7 Accounts use case textual description

Use Case Name	Accounts
Actor	Doctor
Objective	Create new account to other users with appropriate permissions
Precondition	Authentication
Scenario	<ol style="list-style-type: none"> 1.The doctor selects Accounts page 2.The doctor enters new username and password 3.The doctor enters permissions 4. The doctor enters the permission of that user 5. doctor click Save button to save changes
Alternative	None

3.3.6.6 Statistics Use Case

Table 3-8 Statistics use case textual description

Use Case Name	Statistics
Actor	Doctor
Objective	Show statistics
Precondition	Authentication
Scenario	1.The doctor selects Statistics page 2.The doctor enters the year wanted or selects all statistics button 3.The system shows statistics
Alternative	None

3.3.7 Receptionist use case diagram design

The figure below shows use case diagram of the receptionist:

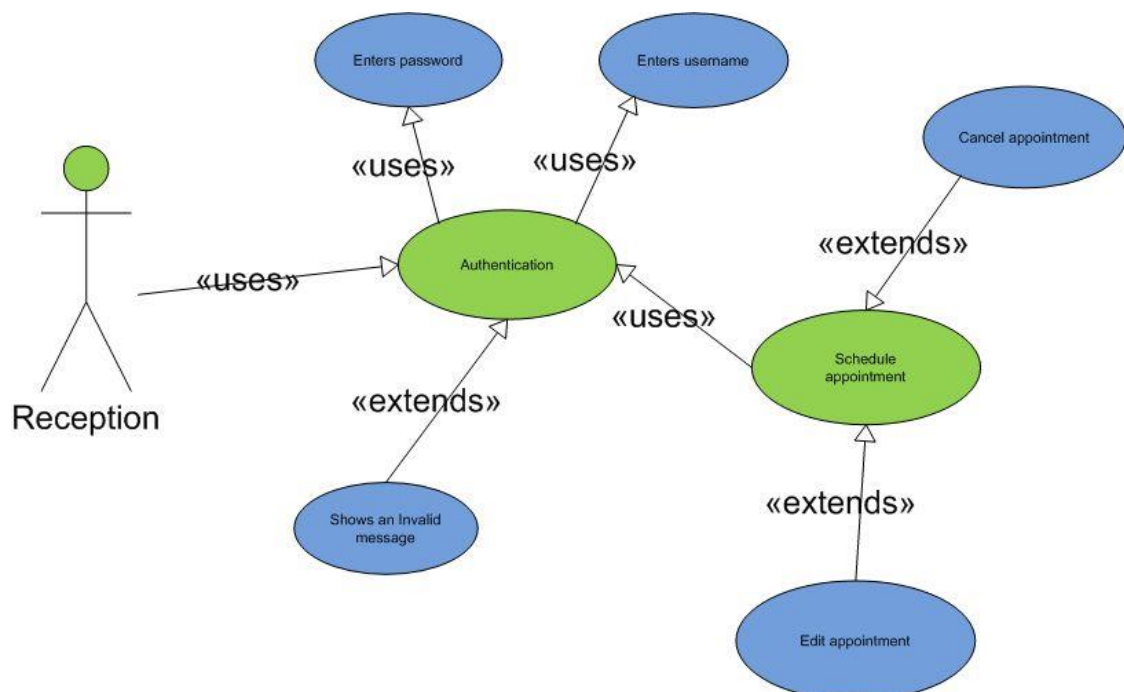


Figure 3-2 Receptionist use case diagram

3.3.8 Receptionist's use case diagram textual description

In the following tables we describe the scenario performed by the receptionist during his/her interaction with the application.

3.3.8.1 Authentication Use Case

Table 3-9 Receptionist authentication use case textual description

Use Case Name	Authentication
Actor	Receptionist
Objective	Login and have access to the application
Precondition	Connect to SQL server
Scenario	<ol style="list-style-type: none"> 1.The receptionist lunches the application 2.The System prompts the receptionist for user name and password 3.The receptionist enters his username and password 4. The system checks the conformity of the information entered by sending an authentication query to the local server 5. The server verifies the query and sends favorable answer. 6. The user accesses the application.
Alternative	If the username or password is wrong or missed the system displays "Invalid authentication" message (return to 2)

3.3.8.2 Schedule Appointment Use Case

Table 3-10 Schedule appointment use case textual description

Use Case Name	Schedule appointment
Actor	Receptionist
Objective	Edit or cancel appointment
Precondition	Authentication

Scenario	<ol style="list-style-type: none"> 1. The receptionist consults table of appointment 2. The receptionist selects patient 3. The receptionist edit appointment 4. The receptionist cancel appointment 5. The receptionist save appointment
Alternative	If the appointment does not need to be edited the receptionist cancel it as soon as he/she reminds the patient

3.4 Class diagram

3.4.1 Definition

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. [4].

A UML class diagram consists of:

- A set of classes
- A set of relationships between classes

3.4.2 Class Definition

A description of a group of objects all with similar roles in the system, which consists:

- Attributes describe what object of the class “know”.
- Operations that defines what objects of the class “can do”.

3.4.3 Visibility of Class attributes and Operations

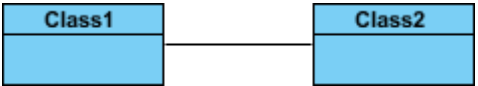
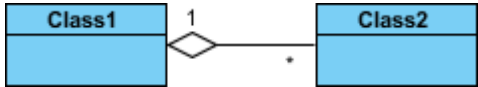
- + denotes public attributes or operations.
- - denotes private attributes or operations.
- # denotes protected attributes or operations.

- ~ denotes package attributes or operations.

3.4.4 Class relationship

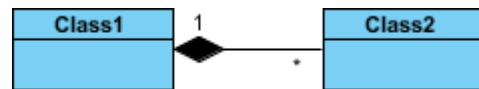
A class may be involved in one or more relationships with other classes. In our class diagram, we have used the following relationships:

Table 3-11 Relationship between two classes

Relationship Type	Graphical Representation
<p>Simple Association:</p> <ul style="list-style-type: none"> • A structural link between two peer classes. • There is an association between Class1 and Class2 • A solid line connecting two classes 	 <pre> classDiagram Class1 --- Class2 </pre>
<p>Aggregation:</p> <p>A special type of association. It represents a "part of" relationship.</p> <ul style="list-style-type: none"> • Class2 is part of Class1. • Many instances (denoted by the *) of Class2 can be associated with Class1. • Objects of Class1 and Class2 have separate lifetimes. • A solid line with a unfilled diamond at the association end connected to the class of composite 	 <pre> classDiagram Class1 o-- "*" Class2 </pre>

Composition:

- A special type of aggregation where parts are destroyed when the whole is destroyed.
- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite



3.4.5 Multiplicity

How many objects of each class take part in the relationships and multiplicity can be expressed as:

Table 3-12 Multiplicity indicator and its meaning

Indicator	Meaning
1	Exactly one
0..*	Many
1..*	One or more

3.4.6 Application class list

The following table lists each class of our application with its related attributes and methods:

Table 3-13 User class description

Class Name: User	
Attributes	Type
u_id	integer

u_username	string
u_password	string
u_name	string
u_permission	integer
addUser() editUser() deleteUser() saveUser() verifyLogin()	

Table 3-14 Patient and Anamnesis class description

Class Name: Patient		Class Name: Anamnesis	
Attributes	Type	Attributes	Type
p_id	Integer	anam_id	Integer
p_fname	String	anam_date	DateTime
p_lname	String	anam_BCG_P	Boolean
p_sexe	String	anam_DTCOQ_3M	Boolean
p_birthdate	DateTime	anam_DTCOQ_4M	Boolean
p_registerdate	DateTime	anam_DTCOQ_5M	Boolean
p_fathername	String	anam_DTCOQ_18M	Boolean
p_mothername	String	anam_DTA_11A	Boolean
p_address	String	anam_DTA_16A	Boolean
p_mobile	Integer	anam_ROUGEOLE_9M	Boolean
p_mchidnb	Integer	anam_DTC_P_R_6A	Boolean
p_fchildnb	Integer	anam_VitamD_1M	Boolean
p_matricule	integer	anam_VitamD_6M	Boolean
		anam_TABAC	Boolean
		anam_bLocation	String
		anam_maternel	String
		anam_feeding	String
		anam_artificialFeeding	String

		anam_reaInterval anam_reaTreatment anam_weight anam_birthTime anam_matInterval	String String Double DateTime String
addPatient() editPatient() deletePatient() selectPatient() searchPatient()		addAnamnesis() editAnamnesis() deleteAnamnesis() saveAnamnesis() selectAnamnesis()	

Table 3-15 Examination,Treatment and FatherJob class description

Class Name: Examination		Class Name: Treatment		Class Name: FatherJob	
Attributes	Type	Attributes	Type	Attributes	Type
exam_id	Integer	treat_id	Integer	Job_id	Integer
exam_date	DateTime	treat_date	DateTime	Job_name	String
exam_age	integer				
exam_size	double				
exam_pc	double				
exam_weight	double				
addExamination() editExamination() deleteExamination() saveExamination() selectExamination()		addTreatment () editTreatment () deleteTreatment () selectTreatment ()		addFatherJob () editFatherJob () deleteFatherJob is() saveFatherJob () selectFatherJob()	

Table 3-16 MotherJob,Appointment,Analysis class description

Class Name: MotherJob		Class Name: Appointment		Class Name: Analysis	
Attributes	Type	Attributes	Type	Attributes	Type
Job_id	integer	ap_id	Integer	analysis_id	integer

Job_name	string	ap_date	DateTime	analysis_name	string
addExamination() editExamination() deleteExamination() saveExamination() selectExamination()		addAppointment () editAppointment() deleteAppointment () selectAppointment()		addAnalysis() editAnalysis() deleteAnalysis() saveAnalysis() selectAnalysis()	

Table 3-17 AnalysisDetails,Diagnostic and DiagSymptom class description

Class Name: AnalysisDetails		Class Name:Diagnostic		Class Name: DiagSymptom	
Attributes	Type	Attributes	Type	Attributes	Type
aDetails_id	integer	diag_id	Integer	diagSymptom_id	integer
aDetails_name	string	diag_name	string	diagSymptom_name	string
addAnalysisDetails () editAnalysisDetails() deleteAnalysisDetails() saveAnalysisDetails() selectAnalysisDetails()		addDiagnostic () editDiagnostic() deleteDiagnostic() selectDiagnostic()		addDiagSymptom() editDiagSymptom() deleteDiagSymptom() saveDiagSymptom() selectDiagSymptom()	

Table 3-18 VisitReasonn, Letter and Certificate class description

Class Name: VisitReason		Class Name:Letter		Class Name: Certificate	
Attributes	Type	Attributes	Type	Attributes	Type
vReason_id	Integer	letter_id	Integer	certificate_id	integer
vReason_name	string	letter_name	string	certificate_name	string
addVisitReason() editVisitReason() deleteVisitReason() saveVisitReason()		addLetter() editLetter() deleteLetter() selectLetter()		addCertificate() editCertificate() deleteCertificate() saveCertificate()	

selectVisitReason()		selectCertificate()
---------------------	--	---------------------

Table 3-19 Prescription, State and City class description

Class Name: Prescription		Class Name: State		Class Name: City	
Attributes	Type	Attributes	Type	Attributes	Type
presc_id	integer	State_id	Integer	City_id	Integer
presc_name	string	State_name	String	City_name	string
presc_volume	string				
addPrescription()		addState()		addCity()	
editPrescription()		editState()		editCity()	
deletePrescription()		deleteState()		deleteCity()	
savePrescription()		saveState()		saveCity()	
selectPrescription()		selectState()		selectCity()	

3.4.7 Class diagram design

The detailed class diagram of our system is presented in Figure 3-3:

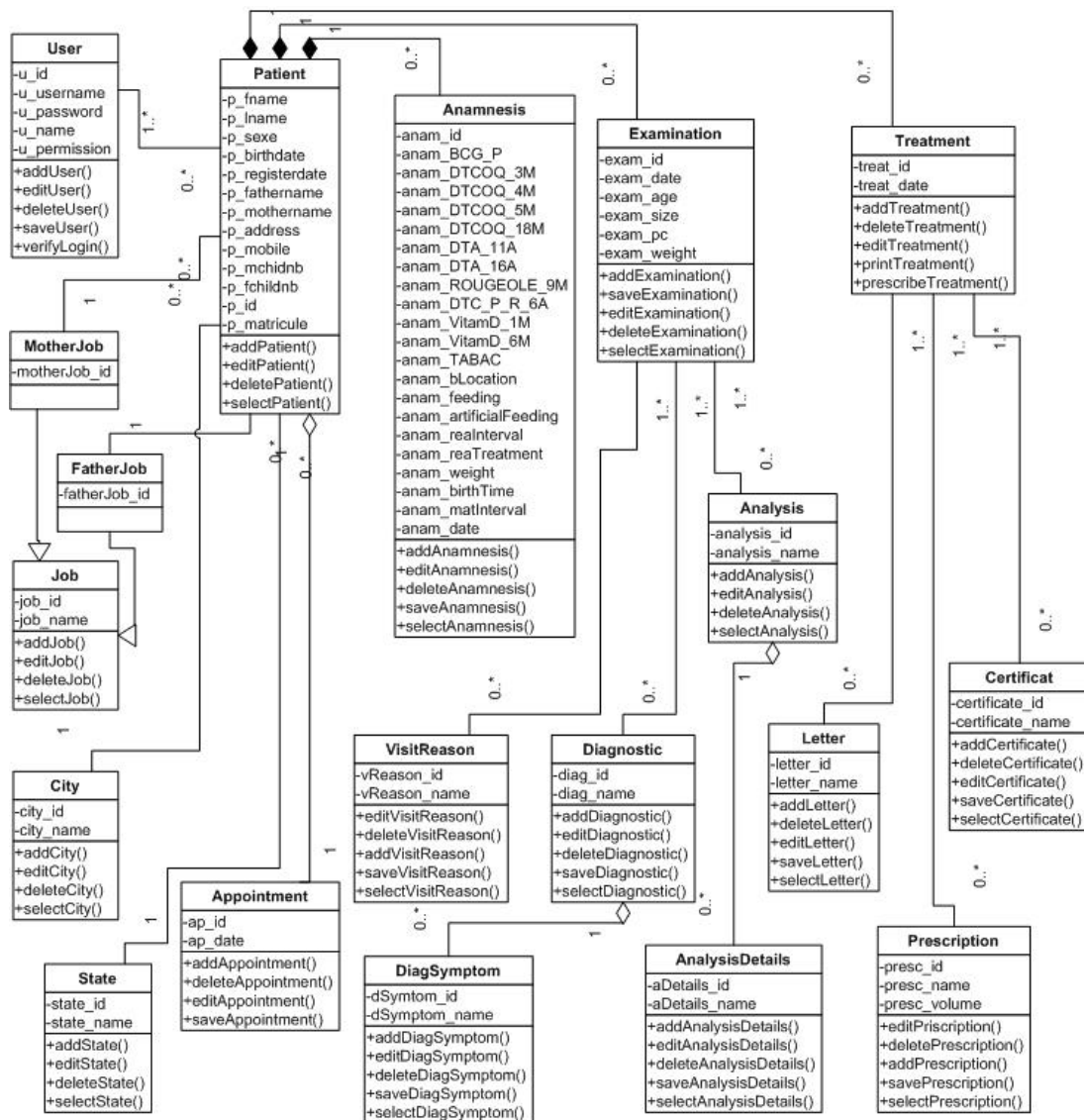


Figure 3-3 Application class diagram

3.5 Sequence diagram

3.5.1 Definition

A sequence diagram is an interaction diagram that details how operations are carried out.

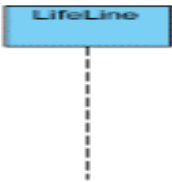

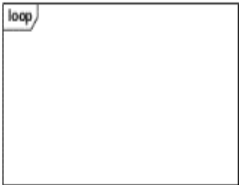
It captures the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by




using the vertical axis of the diagram to represent time, what messages are sent and when.
[16]

3.5.2 Notations

Sequence Diagram Notations are listed in Table 3-20 below with a brief description.


Table 3-20 Sequence diagram notations


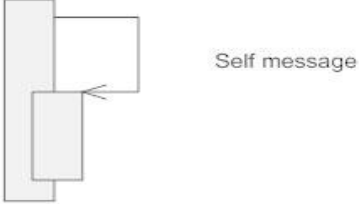
Symbols	Name	Description
	Lifeline Symbol	A sequence diagram is made up of several of these lifeline notations that should be arranged horizontally across the top of the diagram. They represent the different objects or parts that interact with each other in the system during the sequence.
	Activation or execution box	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
	Option loop symbol	Loop fragment is used to represent a repetitive sequence. Place the words 'loop' in the name box and the guard condition near the top left corner of the frame.

	Reference symbol	Ref fragment is used to manage the size of large sequence diagrams. It allows you to reuse part of one sequence diagram in another, or in other words, you can reference part of a diagram in another diagram using the ref fragment.
	Alternative symbol	The alternative combination fragment is used when a choice needs to be made between two or more message sequences. It models the “if then else” logic.
	Actor symbol	Actors are used to represent the users of system; actors can actually be anything that needs to exchange information with the system.

Moreover arrows are used to symbolize the interaction between objects and how the information is transmitted. Table 3-21 lists types of arrows that are used in our sequence diagram:

Table 3-21 Sequence diagram arrow types

Symbol	Name	Description
	Synchronous message	A synchronous message requires a response before the interaction can continue. It's usually drawn using a line with a solid arrowhead pointing from one object to another

	Reply or return message	A reply message is drawn with a dotted line and an open arrowhead pointing back to the original lifeline
	Self message	A message an object sends to itself, usually shown as a U shaped arrow pointing back to itself.

Sequence diagram can not be created for all possible scenarios that may occur in the system. However, we have chosen some of the most important ones done by the Doctor, which are described in the following subsections.

3.5.3 Sequence diagram 1: Authentication

Is the door of the application that means if the user has entered a username and password that correspond to the data stored before in the database the access to the application will be successful, however if one of them was wrong or missed the login will fail.

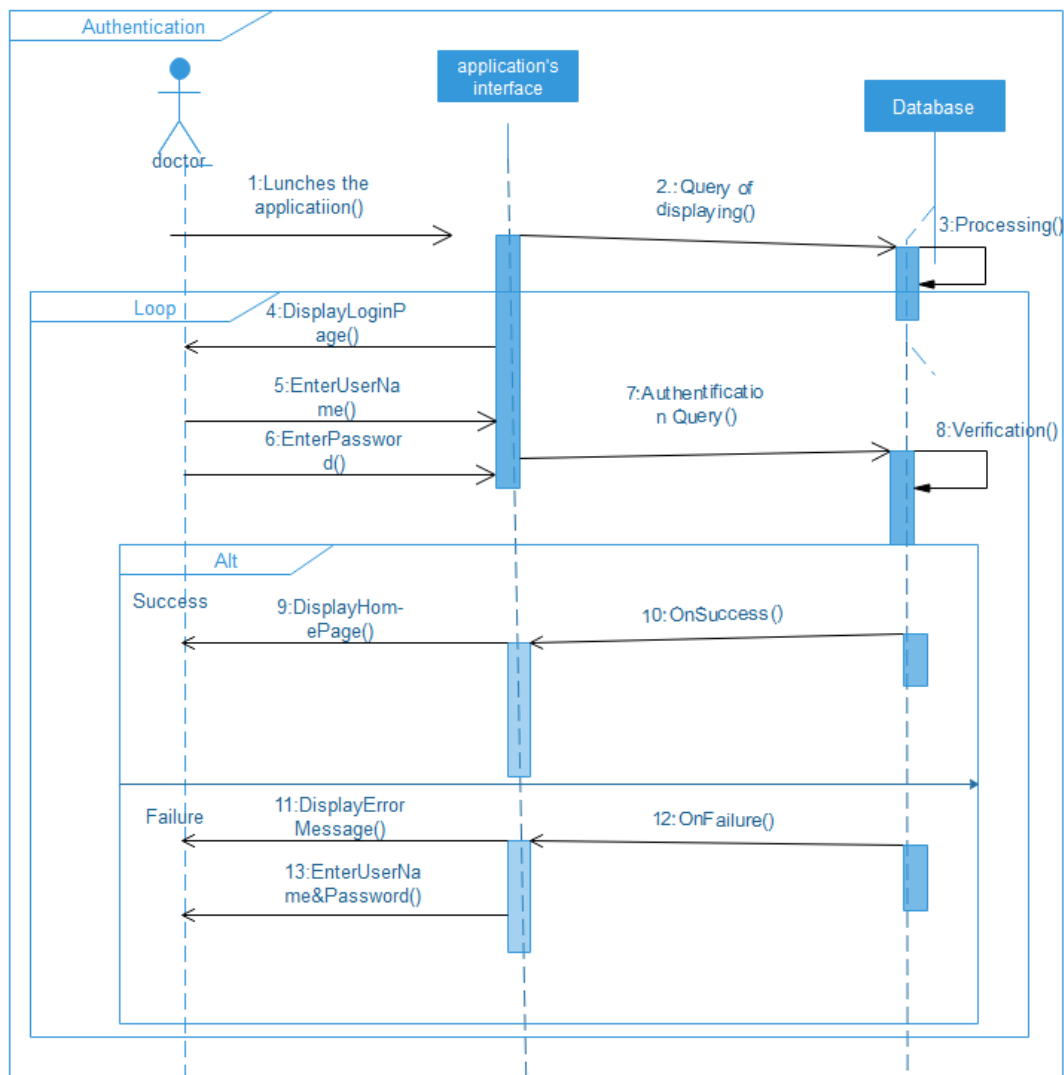


Figure 3-4 Authentication sequence diagram

3.5.4 Sequence diagram 2: Search page (or add patient)

The first interface after accessing to the application is the Search page where the doctor has the possibility to search for an old patient, or selecting him directly from the global list.

In addition, if the patient was new that means he was not registered before, the doctor can make a quick add to that patient.

Both sequences take the doctor to the civil status page to fill it with information related to this patient.

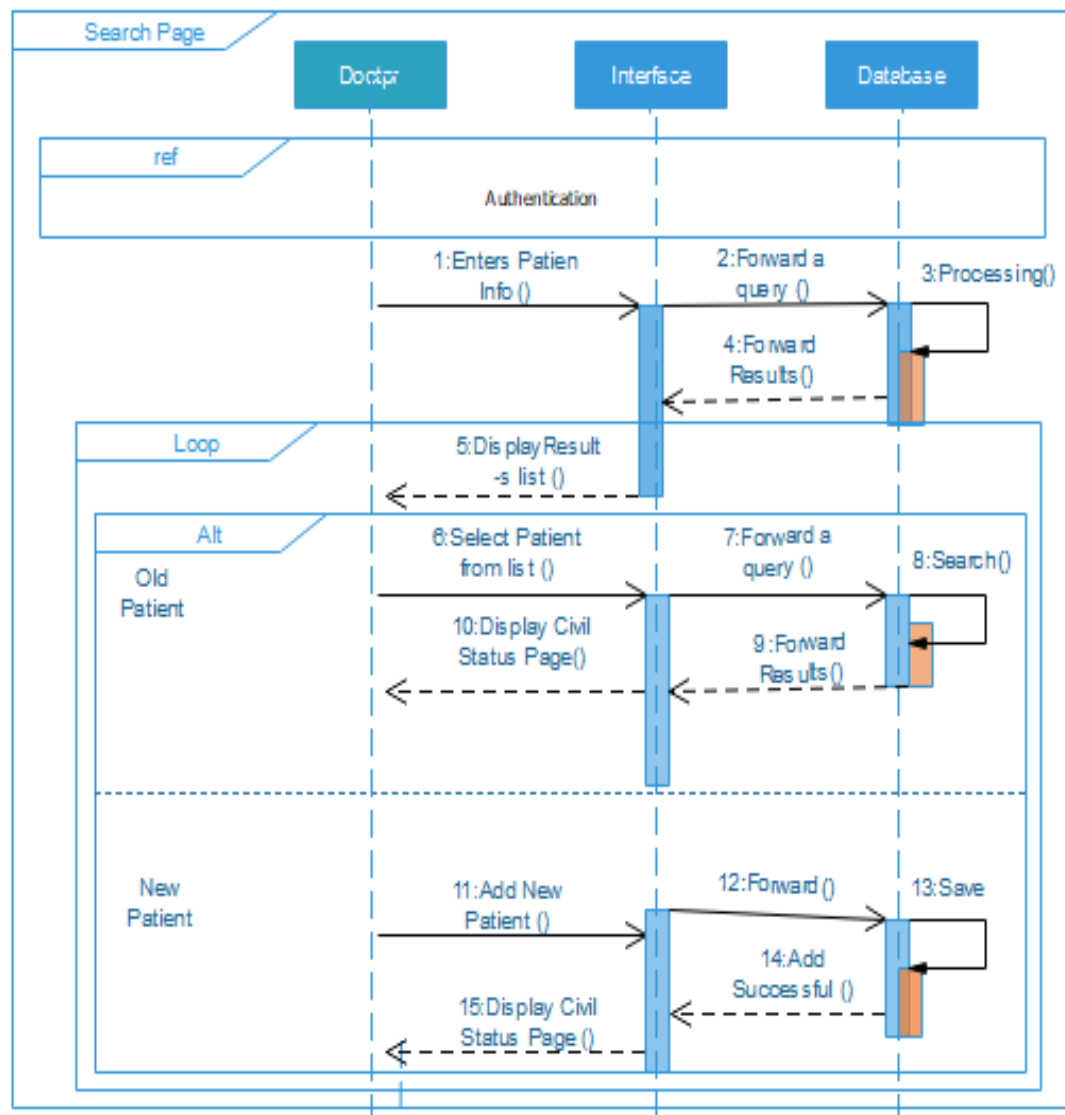


Figure 3-5 Search page sequence diagram

3.5.5 Sequence diagram 3: Anamnesis page

Of course, the doctor can access to anamnesis page after authentication, where he may enter appropriate information for the patient by either selecting or filling blanks.

The save button is responsible for sending query to database that will be finally refreshed, by saving the changes made.

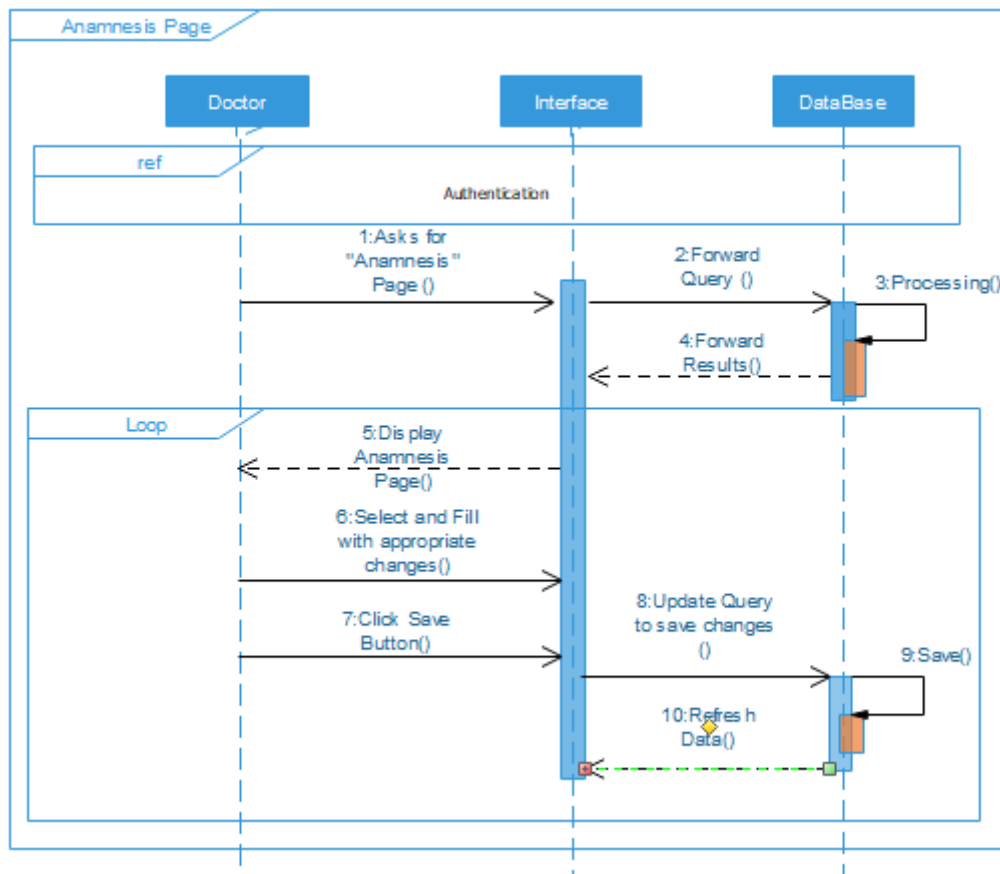


Figure 3-6 Anamnesis page sequence diagram

3.5.6 Sequence diagram 4: Examination page

Also after authentication the doctor selects the Examination page in order to fill it with the appropriate data corresponding to the chosen patient (Reason of visits, symptoms etc. ...).

Here the doctor has the choice either to select items from the list shown in the page or directly enter data which may not exist on the list, in such case the interface shows the doctor a message that allows him to add the inexistent item in the database where will be stored permanently .

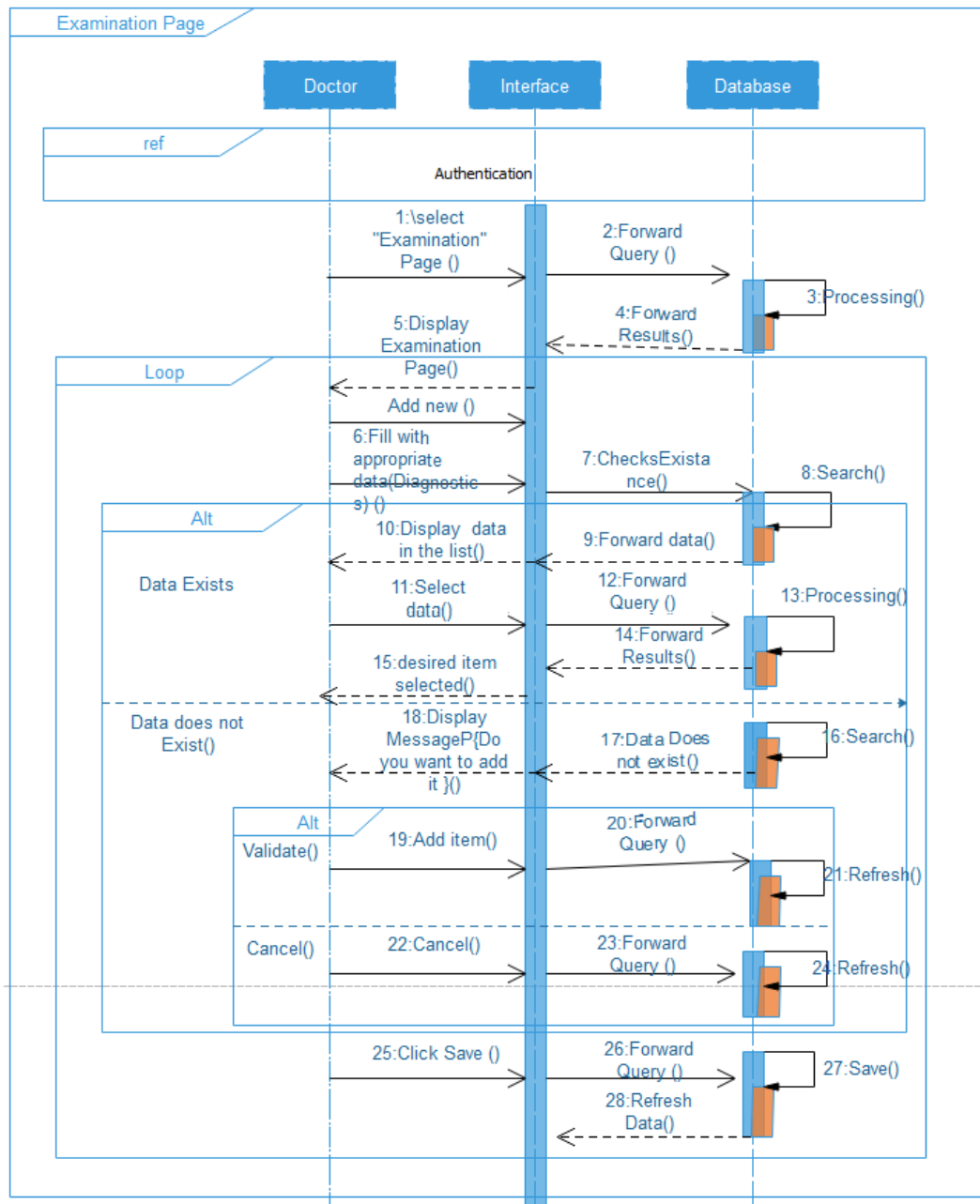


Figure 3-7 Examination page sequence diagram

3.5.7 Sequence diagram 5: Treatment page

Like the other sequences an authentication must occur first, then the doctor selects Treatment page to prescribe medicine to patient.

In this page the doctor have the ability to either select from the list shown in the page or add some new medicines to his previous table in the database.

The doctor may also print the prescription related to any patient.

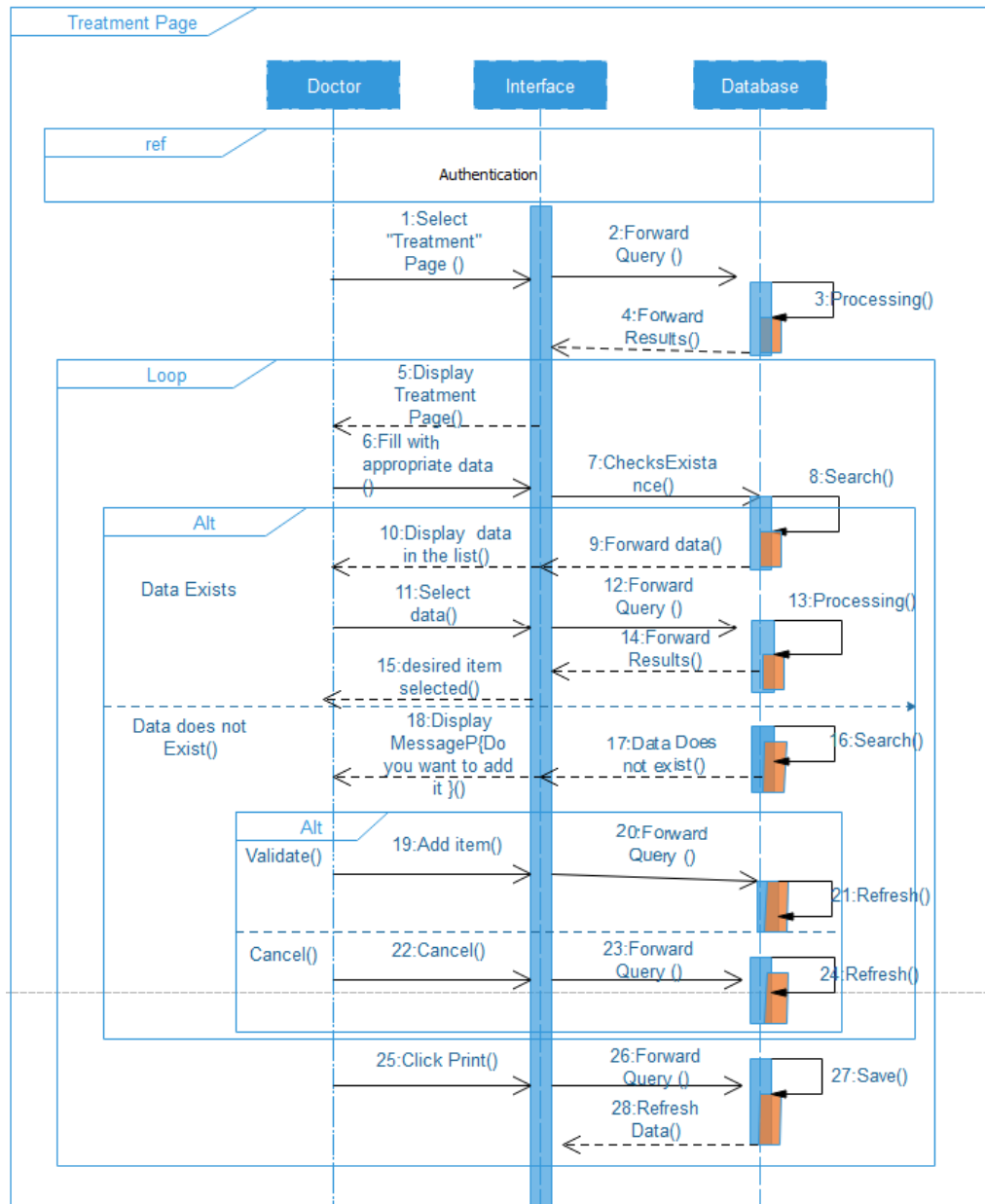


Figure 3-8 Treatment page sequence diagram

3.5.8 Sequence diagram 6: Statistics

Before accessing to statistics, the doctor needs to login, then selects statistics page, where he can either sort them by year or see all statistics directly.

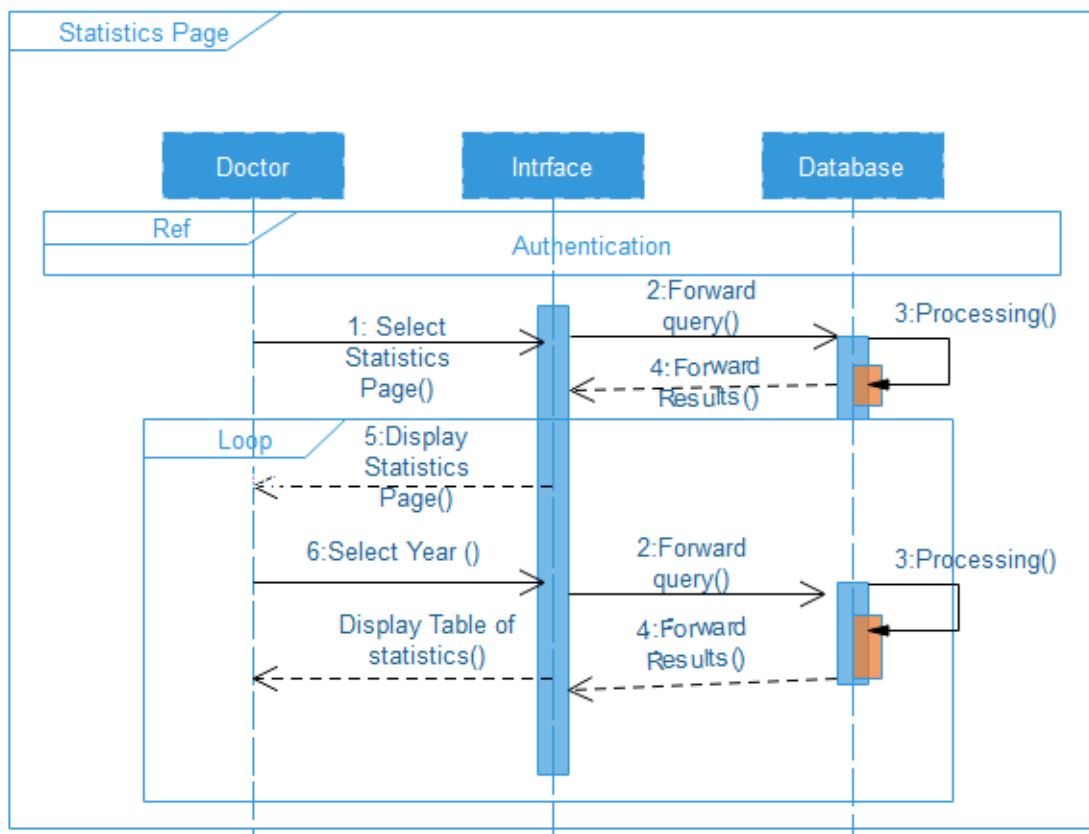


Figure 3-9 Statistics page sequence diagram

3.5.9 Sequence diagram 7: Accounts

In this step and after login, the user selects Accounts page to create an account for another user by entering his username and password, by setting permissions, which let him, access to the pages chosen.

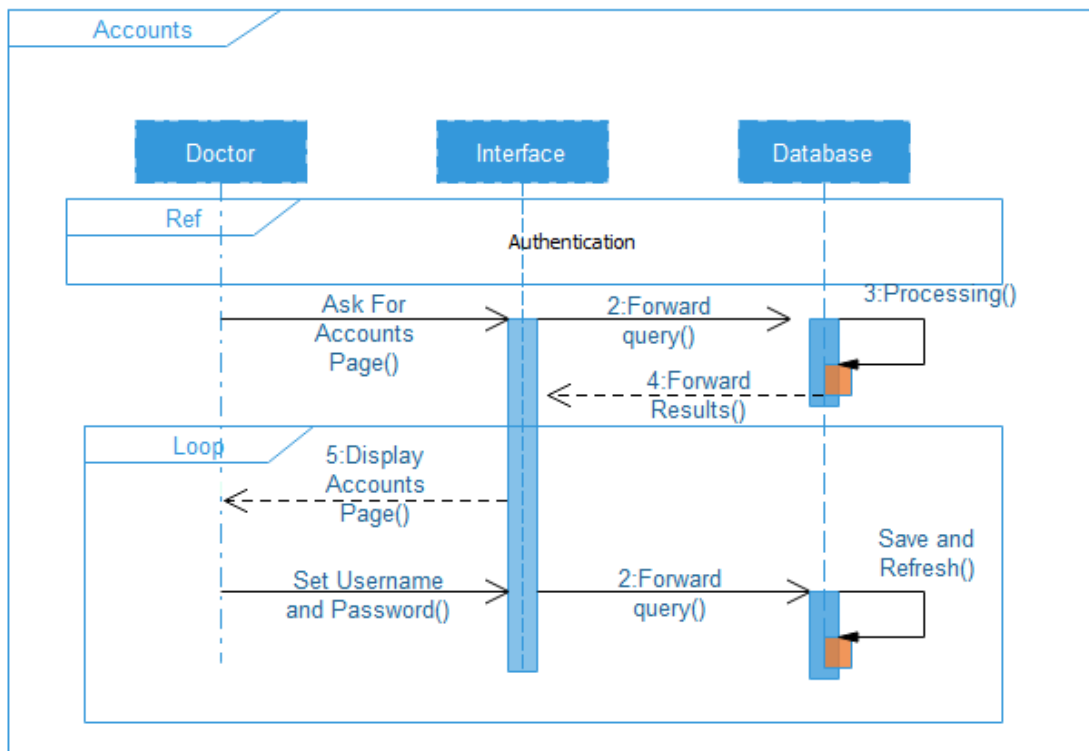


Figure 3-10 Accounts sequence diagram

3.6 Introduction to Database

A database is made up of multiple tables. Just like Excel tables, database tables consist of columns and rows. Each column corresponds to an attribute, and each row corresponds to a single record. Each table must have a unique name in a database. [17]

3.6.1 What is a Relational Database?

Databases can have plenty of tables, and very often, those tables have redundant information. Wouldn't it make sense, both in terms of time saved and in terms of errors avoided, if we could avoid some of that redundancy? Luckily, that is exactly what a relational database does. A relational database allows records from one table to link to related records on different tables. Think of it as a giant spider web. [18]

3.6.2 What is DBMS?

A DBMS is a software that allows creation, definition and manipulation of database, allowing users to store, process and analyse data easily. DBMS provides us with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the database and a lot more. [19]

3.6.3 Database schema

After designing the class diagram of our application, the next step is to transform it into its equivalent database schema as follow:

- First we convert class diagram relationships (one-to-one, one-to-many, many-to-many) into related database tables.
- Define name, attributes, primary keys (**bold**) and foreign keys (#) of each table.
- Define the meanings and type of each attribute.
- Design the schema diagram.

3.7 Convert class diagram relationships

3.7.1 One-to-Many

The one-to-many relations in our class diagram are given in the table bellow:

Table 3-22 One-to-many relationships

Patients - Anamnesis	One patient may have many anamnesis
Patients - Examination	One patient may have many Examination
Patients - Treatment	One patient may have many Treatment
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
Diagnostic - DiagSymptom	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 below:

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.

3.7.2 Many-to-Many

The many-to-many relations in our class diagram are given in the table below:

Table 3-23 Many-to-many relationship

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 – Diagnostic	Many examinations may have the same diagnostics and many diagnostics 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 certificates 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 below:

In a 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 tables Certificat contains (certificat_id, certificat_name).

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

3.8 Database tables

A table is a collection of related data held in a table format within a database. It consists of columns, and rows. [20]

We define our database tables with their attributes as follow:

User(**u_id**,u_username,u_password,u_permission,u_name,u_mobile)

Patient(**p_id**,**p_matricule**,#u_id,p_fname,p_lname,p_gender,p_birthdate,p_registerdate,p_fathername,p_mothername,#p_stateId,#p_cityId,p_address,p_mobile,p_mchidnb,p_fchifnb,#p_fatherJob,#p_motherJob)

Anamnesis(**anam_id**, #p_id, #p_matricule, anam_bLocation, anam_feeding, anam_artificialFeeding, anam_reaInterval, anam_reaTreatment, anam_date, anam_BCG, anam_DTCOQ_3M, anam_DTCOQ_4M, anam_DTCOQ_5M, anam_DTCOQ_18M,anam_DTA_11A,DTA_16A, anam_ROUGEOLE_9M,DTC_P_R_6A, anam_VitamD_1M, anam_VitamD_6M, anam_TABAC, anam_PROPRETE, anam_matInterval, anam_birthmethod, anam_wieght, anam_birthTime).

Treatment(**treat_id**,treat_date,#p_id)

Examination(**exam_id**,#p_id,#p_matricule,exam_date,exam_age,exam_size,exam_pc,exam_wieght)

Appointment(**ap_id**,#p_id,ap_date)

FatherJob(**fatherJob_id**,#job_id)

MotherJob(**motherJob_id**,#job_id)

Job(**job_id**,#job_name)

State(**state_id**,state_name)

City(**city_id**,city_name)

LetterLine(#letter_id,#treat_id)

Letter(**letter_id**,letter_name)

Prescription(**presc_id**,presc_name,presc_volume)

PrescriptionLine(#treat_id,#presc_id)

Certificat(**certificat_id**,certificat_name)

CertificatLine(#certificat_id,#treat_id)

AnalysisLine(#exam_id,#analysis_id)

Analysis(**analysis_id**,analysis_name)

Analysis_Details(**aDetails_id**,#analysis_id,aDetails_name)

DiagnosticLine(#diag_id,#exam_id)

Diagnostic(**diag_id**,diag_name)

DiagSymptom(**dSymptom_id**,dSymptom_name,#diag_id)

VisitReasonLine(#exam_id,#vReason_id)

VisitReason(**vReason_id**,vReason_name)

3.9 Database tables meaning

We describe each table attributes in the following table:

Table 3-24 Database table attributes meaning

Table	attributes	Meaning
User	u_id u_username u_password u_permission u_name u_mobile	User unique identity User username User password Doctor(0),Receptionist(1) User name User mobile number
Patient	p_id p_matricule u_id p_fname p_lname p_gender p_birthdate	Patient unique identity Compose birthYearGenderID Foreign key Patient first name Patient last name Patient gender Patient birth date

	<p>p_registerdate</p> <p>p_fathename</p> <p>p_mothername</p> <p>p_stateId</p> <p>p_cityId</p> <p>p_address</p> <p>p_mobile</p> <p>p_mchidnb</p> <p>p_fchifnb</p> <p>p_fatherJob</p> <p>p_motherJob</p>	<p>Patient first visit date</p> <p>Patient father's name</p> <p>Patient mother's name</p> <p>Patient living state</p> <p>Patient living city</p> <p>Patient address</p> <p>Patient parent phone number</p> <p>Parent number of children m</p> <p>Parent number of children F</p> <p>Patient father's job</p> <p>Patient mother's job</p>
Anamnesis	<p>anam_id</p> <p>anam_bLocation</p> <p>anam_feeding</p> <p>anam_artificialFeeding</p> <p>anam_reaInterval</p> <p>anam_reaTreatment</p> <p>anam_date</p> <p>(anam_BCG,</p> <p>anam_DTCOQ_3M</p> <p>anam_DTCOQ_4M</p> <p>anam_DTCOQ_5M</p> <p>anam_DTCOQ_18M</p> <p>anam_DTA_11A</p> <p>DTA_16A</p> <p>anam_ROUGEOLE_9M</p> <p>DTC_P_R_6A</p> <p>anam_VitamD_1M</p> <p>anam_VitamD_6M)</p> <p>anam_TABAC</p>	<p>Anamnesis identity key</p> <p>Birth location ex Home,hospital</p> <p>Feeding natural or artificial or the two</p> <p>What kind of artificial feeding ex: Lahda</p> <p>Reanimation interval</p> <p>Reanimation treatment</p> <p>Anamneses date of visit</p> <p>Patient Vaccinated or not with these vaccines</p> <p>Patient parent smoking</p>

	anam_PROPRETE anam_matInterval anam_birthmethod anam_weight anam_birthTime p_matricule p_id	Patient cleanliness Maternal interval Normal, cesarean birth Patient weight at birth Before time,in time, after time,normal Patient foreign key Patient foreign key
Treatment	treat_id treat_date p_id	Treatment identity key Treatment date Patient foreign key
Examination	exam_id p_id p_matricule exam_date exam_age exam_size exam_pc exam_weight	Examination unique identity number Patient id Patient matricule Examination date Age at examination Size at examination perimetre cranien at examination patient weight at examination
Appointment	ap_id p_id ap_date	Appointment id Patient foreign key Appointment date when the next session
FatherJob	fatherJob_id job_id	Patirent's father job
MotherJob	motherJob_id job_id	Patient's mother job
Job	job_id job_name	Job unique identity number
State	State_id State_name	State identity State name
City	City_id City_name	City identity City name

LetterLine	letter_id treat_id	Foreign key Foreign key
Letter	letter_id letter_name	Letter unique identity Letter description
Prescription	presc_id presc_name presc_volume	Prescription unique identity Medicine name Medicine volume
Certificat	certificat_id certificat_name	Certificat unique identity Certificat description
CertificatLine	certificat_id treat_id	Foreign key Foreign key
AnalysisLine	exam_id analysis_id	Foreign key Foreign key
Analysis	analysis_id analysis_name	Analysis unique id Analysis ex: echography
Analysis_Details	aDetails_id analysis_id aDetails_name	Primary key Foreign key Analysis details specify which kind of analysis
DiagnosticLine	diag_id exam_id	Foreign key Foreign key
Diagnostic	diag_id diag_name	Diagnostic unique identity Desease type
DiagSymptom	dSymptom_id dSymptom_name diag_id	Diagnostic Symptom unique identity Disease symptom to each disease Foreign key
VisitReasonLine	exam_id vReason_id	Foreign key Foreign key
VisitReason	vReason_id vRreason_name	Patient visit reason Reason of visit ex: control

3.10 Relational database schema design

The following Figure 3-11 shows our database schema:

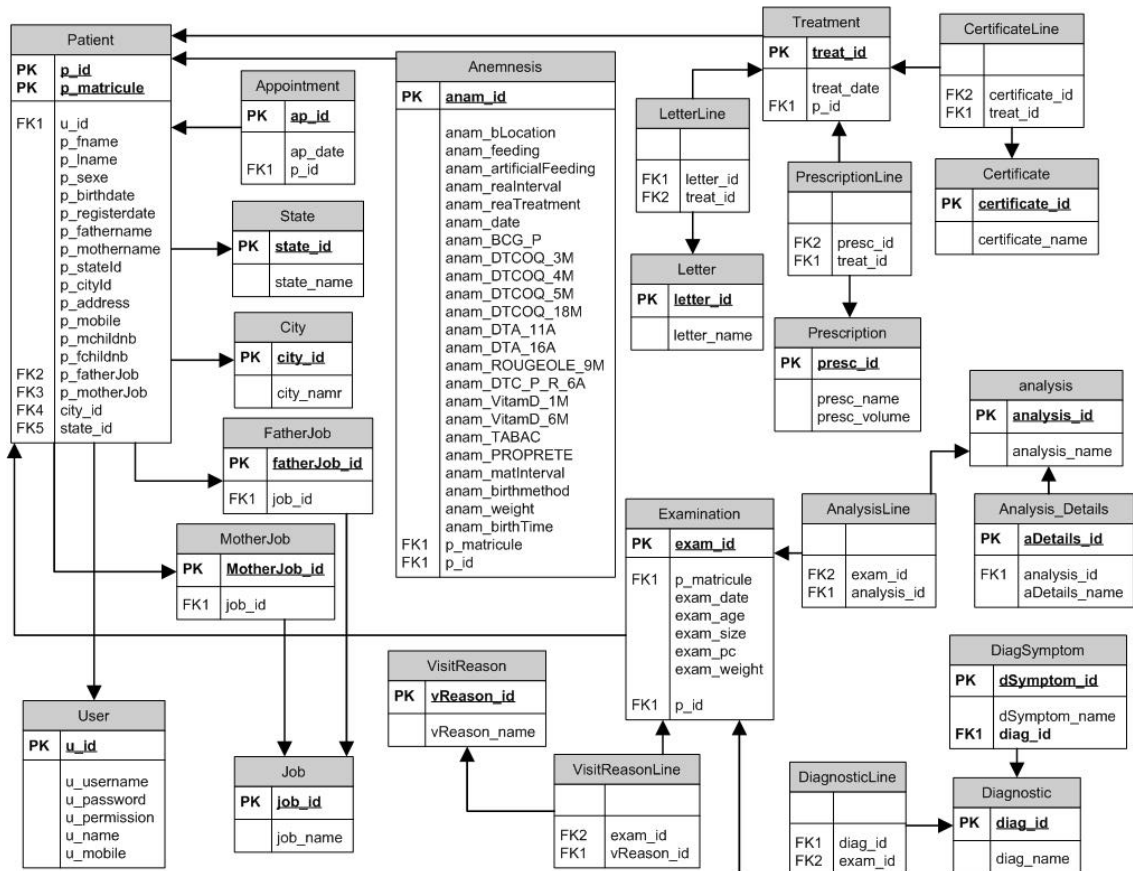


Figure 3-11 Database schema

3.11 Conclusion

In this chapter, we have used features of UML. In addition, we have showed the parts of UML that we have used. Moreover, we modelled our application by providing answers to our modelling and design, based on the analysis of needs of our application. We modelled the app in use case diagram then sequence, and then we designed our application as a class diagram. Finally, we introduced the relational model and the RDBMS; then we convert the class diagram into its related database schema.

Chapter 4: Implementation

4.1 Introduction

This chapter includes a presentation of the different interfaces that our application contains, with a detailed description of the functionalities belonging to each interface. It is to be noted that GUI displaying language is in French as requested by the Doctor.

4.2 Interface of "Authentication" page

To secure and limit the usage of any user, the authentication form will be the first window displayed when the application executes, which prompts the user to enter a valid combination of his username and password.

However, in the case of entering invalid combination or missing a field, a message box shows "Invalid authentication".

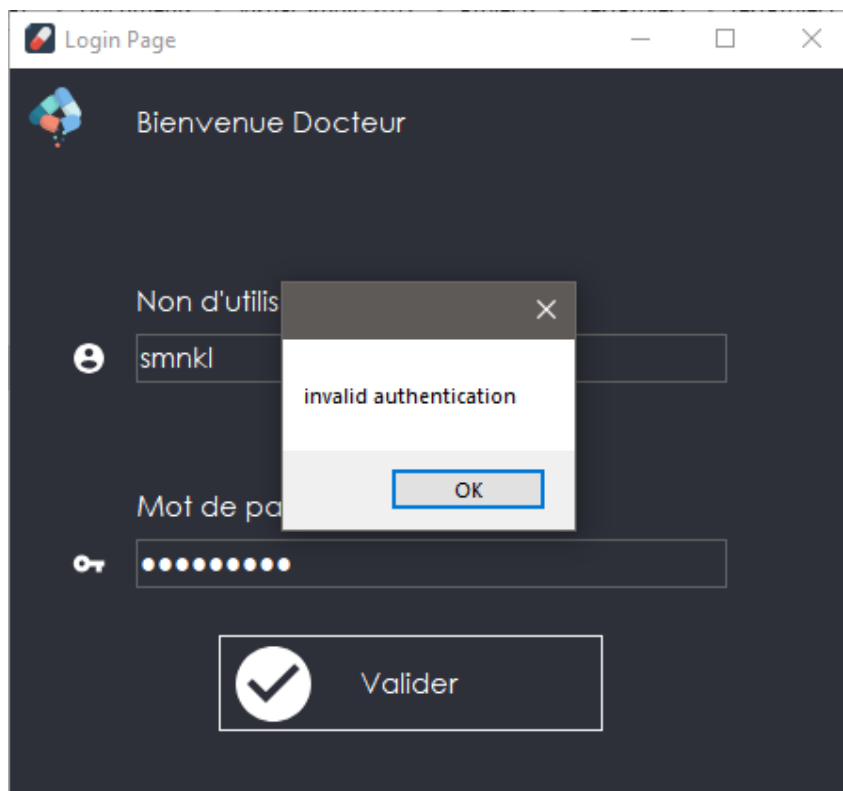


Figure 4-1 Interface of authentication page

4.3 Interface of the "Doctor Home page"

After the doctor login with valid credential, the home window form shows up, and this form contains four main pages: search, tools, accounts and statistics.

It is obvious that the doctor needs to search for patient's record as soon as the patient enters, for that reason, the search page is the first window displayed.

This page contains a search engine where the doctor can look for any patient by entering either his (first name, last name, state, city, or matricule) then clicks on "Commencer la recherche" to start searching .

The doctor then selects the patient from the table in the bottom of the page and either clicks "Selecter" button to select him or "Supprimer" button to delete him from the database.

Nom	Prénom	Commune	Sexe	Wilaya	Matricule	Date de Naissance
	IBTISSEM	Cap Djinet	M	CANADA	2009F1	08/02/2009
	MOUNIR	Figuer	M	BOUMERDES	2009M2	24/01/2009
	ABDELAH	Boumerdes	M	BOUMERDES	2002M3	01/10/2002
	ABDELMALEK	Boumerdes	M	BOUMERDES	2002M4	01/10/2002
	ANIS	Cap Djinet	M	BOUMERDES	2006M5	09/08/2006
	BOUCHRA	Alligula	F	BOUMERDES	2009F6	28/02/2009
	Med YACOB	Legata	M	BOUMERDES	2009M7	19/01/2009

Figure 4-2 Interface of doctor's home page

4.4 Interface of the "Receptionist Home page"

When the receptionist login by entering his username and password, the previous page will appear.

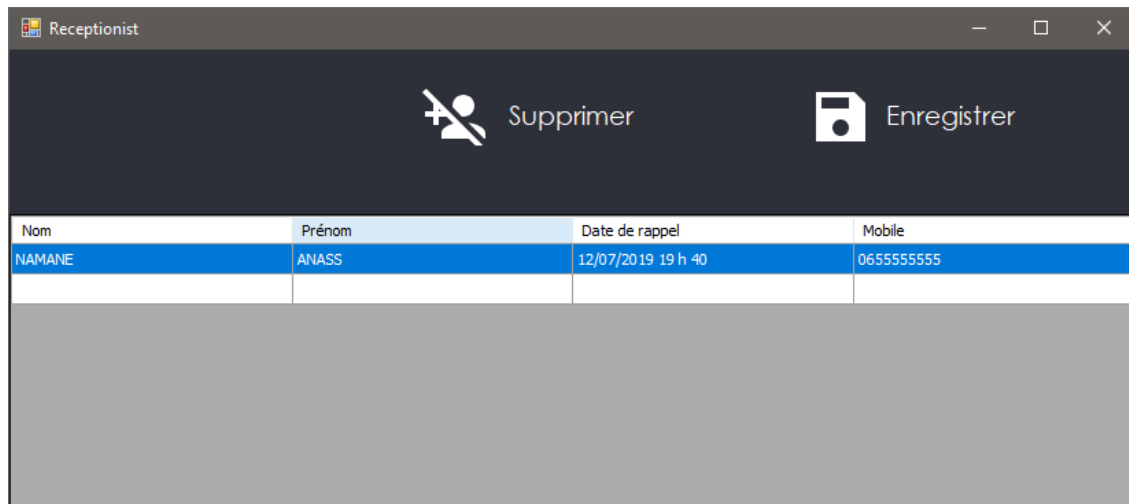


Figure 4-3 Interface of receptionist home page

The receptionist has the ability either of deleting an appointment by clicking on "Supprimer" button or editing the date of any patient by clicking on it in the table shown on the bottom of the page. At the end, the receptionist clicks on "Enregistrer" button to save changes made.

4.5 Interface of "Add a patient" page

When the doctor receives a new patient who does not have a stored record before, the doctor needs to add him by clicking "Ajouter" button, and fill the fields of "Ajouter un nouveau patient" window with the appropriate information related to this patient. Finally, the doctor clicks on "Valider l'ajout" to validate the adding.

Figure 4-4 Interface of add patient page

4.6 Interface of “Outils” page

The doctor selects this page in order to either add, delete or modify the tabs shown on the top of the window " objet du control, lettres, ordonances, professions, certificats " which will be used in other pages further .

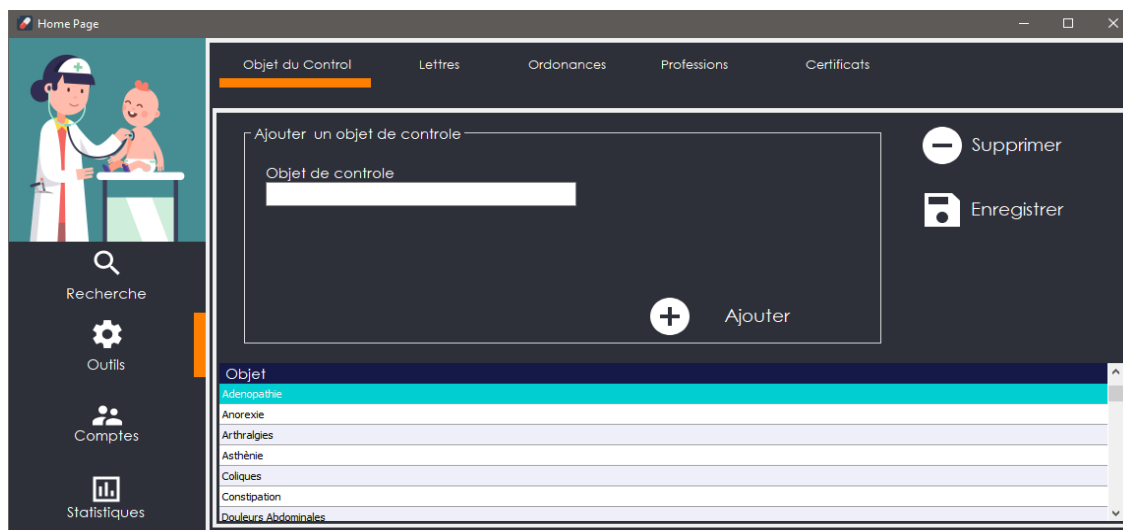


Figure 4-5 Interface of tools page

4.7 Interface of "Comptes" page

The doctor selects this page for the reason of creating an account for another user either receptionist or another doctor. The doctor needs to specify a username, password corresponding to that user, and permissions based upon the role of e "0" for the receptionist and "1" for the doctor.

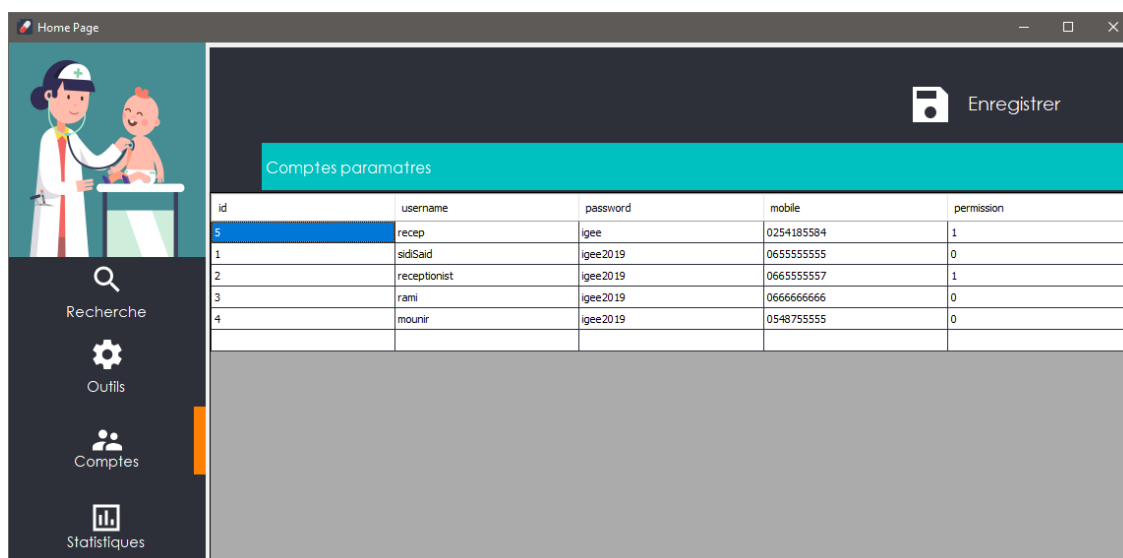


Figure 4-6 Interface of accounts page

4.8 Interface of "Etat Civil" page

As soon as the doctor clicks on "Valider l'ajout" after adding a new patient or in the case of selecting an old patient on the list shown in the Search interface, this page appears filled with patient information entered before.

This page contains information corresponding to both patient and parents identity.

The doctor fills the fields then clicks "Valider" button to save changes or store data.

"Annuler" button is used to cancel the modification made.

"A rappeler" button is used to send the information of the concerned patient Receptionist page where will be added in the list.

Figure 4-7 Interface of civil status page

4.9 Interface of "Anamnese" page

In this page, the doctor selects the medical background related to one patient (vaccines and other ones).

The patient may have many anamnesis pages for that reason the doctor is able to delete previous, add new pages.

Finally, the doctor save the content by clicking on "save icon" on the top of the page.

Figure 4-8 Interface of anamnesis page

4.10 Interface of "Examen" page:

This page is selected in order to get firstly the reason of the visit, and then identify the illness according to symptoms and diagnostics.

We have for each diagnostic some related symptoms and the same thing each analysis consists of its own details.

Either the doctor has the possibility to fill the fields by entering manually or to select the item wanted from the comboBox appeared in the right of the page

Figure 4-9 Interface of examination page

4.11 Interface of "traitements" page

After the doctor identifies the disease, he moves to treatment page to prescribe medicine for the patient.

The doctor selects prescription "ordonnance" in "Type de modèle" comboBox then selects from the "description" comboBox the medicine to be given to the patient, and if he wants to edit the volume, he does so by modifying it on the list shown above.

In addition, if the description comboBox does not contain the medicine wanted to be prescribed, the doctor has the ability to enter this last manually.

Finally, the doctor clicks on "Print" button to print the prescription and also to save both the medicine given and the added ones, or even export it as PDF file and stores in his personal computer.

Patients

Etat Civile Amenese Examen **Traitement** Graph

Nom: NAMANE Prénom: ANASS Matricule: 2010F33604
 Date de naissance: mardi 26 janvier 2010

Type De Modèle: Ordonnances
 Description: AMOXIL 500 gel

⊕ Prescrire Print

Ordonnance	Volume
BRICANYL Spray	1bouffee matin et soir pendant 07 jours
CAMPHOBOTIC Suppo NRS	1 suppo le soir
AMOXIL 500 gel	1gel 3 fois par jour pendant 08 jours

Figure 4-10 Interface of treatment page

4.12 Interface of "Graph" page

This page shows two graphs related to the patient.

The first graph shows the development of his cranial perimeter “périmètre crânien” over his age (in months).

The second graph shows the development of his weight over the time (in months).

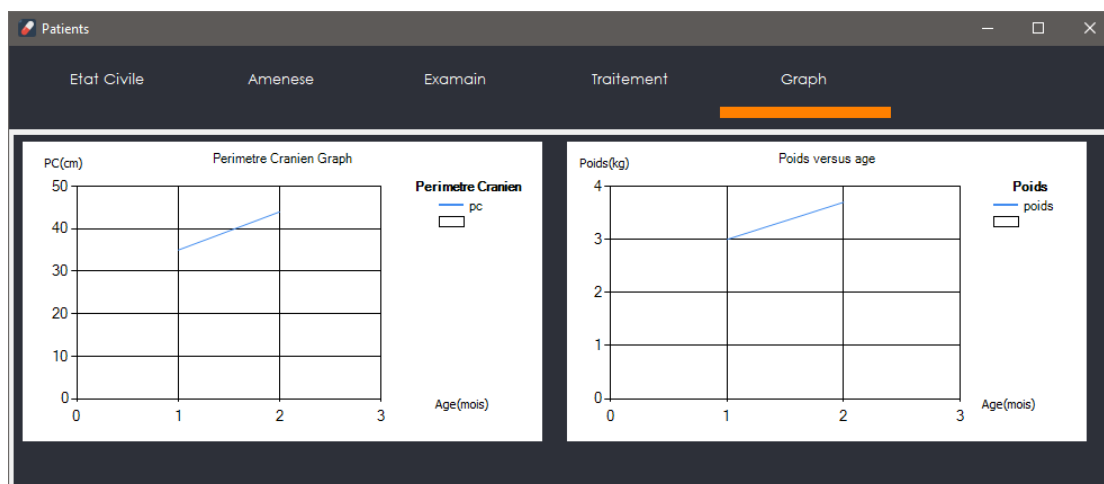


Figure 4-11 Interface of graph page

4.13 Interface of Statistics page

Here in this page which is accessed only by the doctor, shows up some wanted statistics such as the total number of visits etc ...

the doctor then has the possibility of selecting statistics by choosing the year and click on **Selecter** button in addition to that he can also see all statistic at one time by clicking on "Toutes les statistiques".

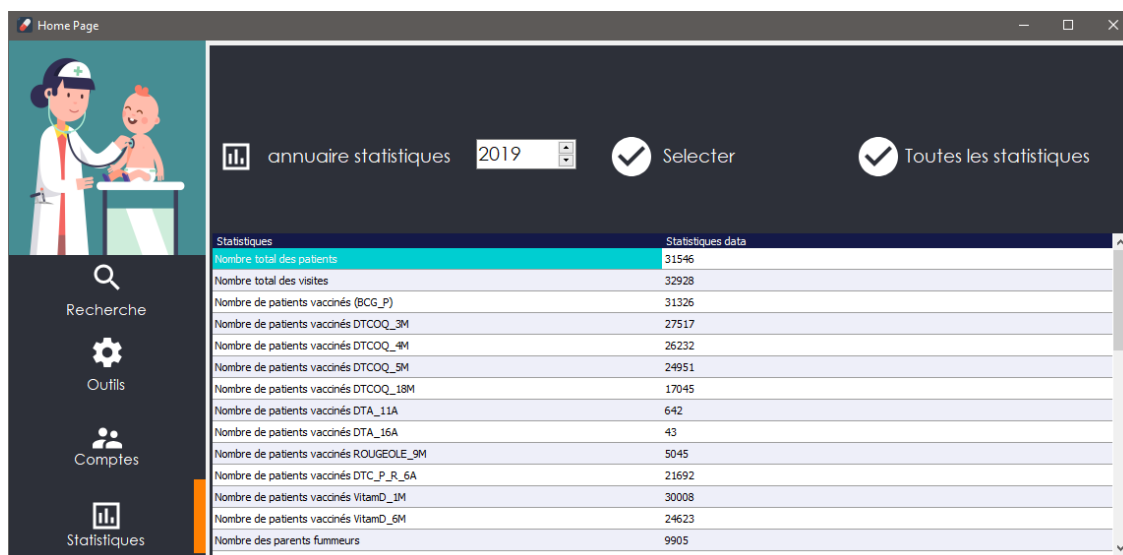


Figure 4-12 Interface of statistics page

4.14 Interface of Receptionist home page

The receptionist has limited access permissions, he/she cannot access to patients records. However, the receptionist have access to the appointments records, his/her main task in our application is to schedule appointments, which allows her/him to select, edit and cancel a patient appointment.

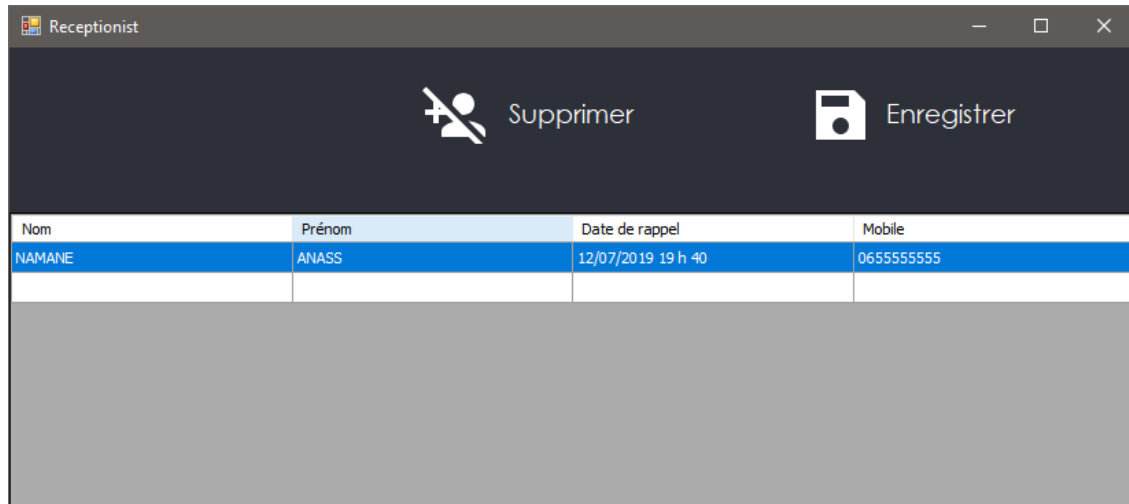


Figure 4-13 Receptionist home page

4.15 Conclusion

In this chapter, the different graphical user interfaces have been presented for both Doctor and Receptionist functionalities with a description for each component within interfaces. According to that description, it is clear that the system actors are able to use our application in a very easy and friendly manner for fulfilling any desired task.

General Conclusion

In this report, we have presented a step-by-step methodology in the design and the implementation of a software management system for pediatrics office. In order to achieve the objectives of our requirement document, we have proceeded by the different stages of the software life cycle including the analysis, design, implementation and the test of the result.

The automation of the different functionalities as requested by the Doctor has been satisfactory achieved. In addition, the implemented application gathers both the ease of use and the rapid access to the data. Furthermore, it provides a reliable system that effectively reduces the loss of time and the tiredness of the doctor.

This project was very interesting to design and implement a software system, because it allowed us to become familiar with new concepts, and to improve our knowledge and skills in the field of object-oriented programming and information system.

As future works, our system can be improved further by adding some extra functionalities among which:

- Billing of patients.
- Building a network of pediatricians.
- Patient queue.
- System backup and recovery management.

Webography

- [1] Microsoft, «Microsoft Docs,» 19 03 2019. [En ligne]. Available: <https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019>.
- [2] Microsoft Docs, «Microsoft Docs,» 12 06 2019. [En ligne]. Available: <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017>.
- [3] «Introduction to C#,» [En ligne]. Available: <https://www.geeksforgeeks.org/introduction-to-c-sharp/>. [Accès le 02 06 2019].
- [4] K. Boyini, «Structured Query Language (SQL),» 23rd Jul, 2018, [En ligne]. Available: <https://www.tutorialspoint.com/Structured-Query-Language-SQL>. [Accès le 02 06 2019].
- [5] rpetrusha, mikkeltu, nschonni, mairaw et nemrism, «.NET Framework Guide,» 02 04 2019. [En ligne]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/>.
- [7] «BindingSource Component Overview,» Docs.microsoft.com, [En ligne]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/winforms/controls/bindingsource-component-overview>. [Accès le 02 06 2019].
- [8] «What is Modeling Language? - Definition from Techopedia,» Techopedia, [En ligne]. Available: <https://www.techopedia.com/definition/20810/modeling-language>. [Accès le 02 06 2019].
- [9] «What is Unified Modeling Language (UML)? - Definition from Techopedia,» [En ligne]. Available: <https://www.techopedia.com/definition/3243/unified-modeling-language-uml>.

- [10] «UML 2.4 Diagrams Overview,» [En ligne]. Available: <https://www.uml-diagrams.org/uml-24-diagrams.html#structure-diagram>. [Accès le 05 02 2019].
- [11] «Use Case Diagrams Online, Examples, and Tools,» Smartdraw.com, [En ligne]. Available: <https://www.smartdraw.com/use-case-diagram/>.
- [12] «Sequence Diagram Tutorial: Complete Guide with Examples - Creately Blog,» Creately Blog, [En ligne]. Available: <https://creately.com/blog/diagrams/sequence-diagram-tutorial/>.
- [13] «What is Class Diagram?,» Visual-paradigm.com, [En ligne]. Available: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/>.
- [14] «What is Use Case Diagram?,» [En ligne]. Available: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>.
- [15] What-is-class-diagram?, «visual-paradigm.com,» 2019. [En ligne]. Available: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/>.
- [16] «What is Sequence Diagram?,» Visual-paradigm.com, [En ligne]. Available: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>.
- [17] «Ditch Your Spreadsheet for a Database to Access the Power of Your Data,» Lifewire, [En ligne]. Available: <https://www.lifewire.com/what-is-a-database>
- [18] Kaitlin et Oglesby, «Relational Database: Model & Example,» study.com, [En ligne]. Available: <https://study.com/academy/lesson/relational-database-model-example.html>.
- [19] «Overview of Databse and DBMS,» Studytonight, [En ligne]. Available: <https://www.studytonight.com/dbms/overview-of-dbms.php>.
- [20] «Table (database),» En.wikipedia.org, [En ligne]. Available: [https://en.wikipedia.org/wiki/Table_\(database\)](https://en.wikipedia.org/wiki/Table_(database)).

Bibliography

- [6] B. Johnson, *Professional Visual Studio 2013*, Indianapolis: Ind.: Wrox, 2014.