People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
M'Hamed Bougara University – Boumerdès



Faculty of Science
Department of Computer Science

Field : Computer Science
Major : Computer Science
Speciality : Information System (IS)

Bachelor's Thesis

for the obtainment of an Academic Bachelor's Degree

Title :

Development of a Web Application for the Management of Hospitalized Patients

Presented by :
DJIDA Selma
HECHICHE Rania
LALLEM Meriem

Presented on : 03/06/2026, in front of the following jury :

Mr. CHAABANI Mohamed	examiner
Ms. SENOUSSI Nour El-Houda	supervisor

Acknowledgements

We begin by expressing our gratitude to Allah, the Almighty, for granting us the strength, health, and patience needed to complete this work.

We would also like to sincerely thank our supervisor, Ms. SENOUSSI Nour El-Houda, for her guidance, availability, trust, and valuable advice throughout the realization of this project.

We would also like to thank our examiner, Mr. CHAABANI Mohamed, for accepting to dedicate their time to evaluate our work.

A special thanks goes to our families for their constant support, encouragement, patience, and for always believing in us throughout our academic journey.

We would also like to express our sincere thanks to our friends Sara, Lina and Ahlam, Farah and Zineb, Dina, Lydia and Mounira for their support, encouragement, and the beautiful moments shared during this journey. Their collaboration and support enriched our reflection and were of great value throughout our academic path.

Finally, we would like to thank everyone who contributed directly or indirectly to the success of this project by sharing their knowledge, offering their help, and supporting us throughout this academic experience.

May everyone who played a role in our journey be rewarded and blessed.

Résumé

Ce mémoire présente le développement d'une application web de gestion des informations des patients dans les hôpitaux. Cette solution vise à centraliser les données médicales et à simplifier le travail du personnel soignant en facilitant la consultation, la mise à jour et le suivi des dossiers des patients.

L'application améliore l'organisation et l'efficacité de la gestion hospitalière grâce à une interface simple et intuitive. Elle a été développée à l'aide de technologies web modernes, notamment HTML, CSS, JavaScript, Node.js et Express.js.

Mots clés : Application web, gestion hospitalière, patient, base de données, Node.js, Express.js, interface utilisateur.

Abstract

This thesis presents the development of a web application for managing patient information in hospitals. The system aims to centralize medical data and simplify the work of healthcare staff by enabling easy access, updating, and monitoring of patient records.

The application improves hospital management efficiency by providing a simple and user-friendly interface for better organization and data handling. It was developed using modern web technologies such as HTML, CSS, JavaScript, Node.js, and Express.js.

Keywords : Web application, hospital management, patient, database, Node.js, Express.js, user interface.

Table of Contents

List of Figures **I**

List of tables **II**

General Introduction **1**

1 General information about the web **2**
1 Introduction . . . 3
2 Definition of a website . . . 3
3 Types of websites . . . 3
3.1 Static websites . . . 3
3.2 Dynamic websites . . . 3
4 Key differences between static and dynamic websites . . . 4
5 Operating principle of websites . . . 4
5.1 Web server . . . 4
5.2 Web browser . . . 5
5.3 Programming languages used for web development . . . 5
5.3.1 Front end development languages . . . 5
5.3.2 Back end development languages . . . 5
5.4 Database server . . . 5
5.5 Client-server relationship . . . 5
6 Web architectures . . . 6
6.1 1-Tier Architecture . . . 6
6.2 2-Tier Architecture . . . 6
6.3 3-Tier Architecture . . . 6
6.4 N-Tier Architecture . . . 6
7 Conclusion . . . 6

2 Design and Analysis **7**
1 Introduction . . . 8
2 Functional and Non-functional Requirements . . . 8
2.1 Functional Requirements . . . 8
2.2 Non-functional Requirements . . . 8
3 UML Definition . . . 9
4 Detailed System Description . . . 9
5 Use Case Diagram . . . 9
5.1 Definition . . . 9
5.2 Identification of Actors . . . 9
5.3 Description of Interactions . . . 10
5.4 General Use Case Diagram . . . 12
6 Sequence Diagram . . . 12
6.1 record vital signs Sequence Diagram . . . 13
6.2 creation user acount Sequence Diagram . . . 14

6.3 Appointment Scheduling Sequence Diagram 15
7 Class Diagram 15
8 Relational Model 17
8.1 Definition 17
8.2 Rules for Converting a Class Diagram 17
8.3 Relational Schema 17
9 Conclusion 18

3 System Implementation **19**
3.1 Introduction 20
3.2 Programming languages 20
3.2.1 HTML 20
3.2.2 CSS 20
3.2.3 JavaScript 20
3.2.4 Node.js 20
3.2.5 Express.js 21
3.3 Development tools 21
3.3.1 MySQL 21
3.3.2 Git 21
3.3.3 GitHub 21
3.3.4 Visual Studio Code 22
3.3.5 Draw.io 22
3.4 Presentation of the application's interfaces 22
3.4.1 Create Stuff Account Page 22
3.4.2 Medical Secretary Dashboard 23
3.4.3 Registration Page 23
3.4.4 Schedule Appointment Page 24
3.5 Conclusion 24

General Conclusion **25**

List of Figures

2.1 General Use Case Diagram 12
2.2 Sequence Diagram – vital signs record 13
2.3 Sequence Diagram – acount creation 14
2.4 Sequence Diagram – appointment scheduling 15
2.5 Class Diagram of the System 16

3.1 HTML5 logo [17] 20
3.2 CSS3 logo [19] 20
3.3 JavaScript logo [21] 20
3.4 Node.js logo [23] 21
3.5 Express.js logo [25] 21
3.6 MySQL logo [27] 21
3.7 Git logo [28] 21
3.8 GitHub logo [31] 21
3.9 Visual Studio Code logo [33] 22
3.10 Draw.io logo [35] 22
3.11 Create Account page 22
3.12 Schedule Appointment Page 23
3.13 register new patient page 23
3.14 Schedule Appointment Page 24

List of Tables

1.1 Comparison between static and dynamic websites [4]. 4

2.2 Actor interactions with the system 11

General Introduction

Hospitals and medical centers must manage a large volume of information related to patients, medical treatments, and hospital services. Ensuring that this information is well organized and easily accessible is essential for providing efficient and reliable medical care.

In many hospitals, managing patient data can be challenging due to the large number of patients and the complexity of medical information that needs to be recorded and updated regularly. Without an efficient system, it can become difficult for healthcare professionals to quickly access patient records, follow patient progress, and coordinate medical activities.

The use of modern web technologies offers new opportunities to improve hospital management systems. Web applications allow medical staff to access information quickly, update patient records in real time, and manage healthcare activities more efficiently.

For this reason, this project focuses on the development of a web application designed to assist hospital staff in managing and monitoring patient information. The proposed system aims to simplify daily administrative and medical tasks while improving the organization and accessibility of patient data within the hospital environment.

Problem statement

Managing patient information in hospitals is a complex task that involves handling and updating a large amount of medical and administrative data. When done manually or stored in different places, medical files can be lost, duplicated, or hard to access, which may confuse medical staff, affect hospital organization, and reduce the quality of care provided to patients.

Communication between hospital staff can also become difficult, leading to appointment conflicts or missing information. In addition, manually tracking treatments, patient conditions, and bed occupancy takes time and increases the risk of human error.

Proposed solution

To make hospital management easier and more organized, we developed *MediCare*, a web-based Hospitalized Patients Management System. The application brings all hospital information together in one centralized platform.

With MediCare, medical staff can easily manage patient records, appointments, treatments, and bed occupancy. Doctors can quickly access medical files, nurses can follow patient care and vital signs, and medical secretaries can manage admissions and scheduling more efficiently.

The system helps reduce paperwork, avoid loss or duplication of files, improve communication between departments, save time, and provide better healthcare services for patients.

Chapter1

GENERAL INFORMATION ABOUT THE WEB

1 Introduction

A web application is a website that contains pages with partly or entirely undetermined content. The final content of a page is determined only when the visitor requests a page from the web server. Because the final content of the page varies from request to request based on the visitor's actions, this kind of page is called a dynamic page [1].

In this chapter, we will cover the basic concepts of web applications, key differences between static and dynamic websites, and how web technologies work together to deliver interactive experiences.

2 Definition of a website

A website is a set of interconnected web pages, including multimedia content, typically identified by a common domain name and accessible through a web browser. Websites are hosted on servers and can be accessed via the internet using a URL (Uniform Resource Locator). They are the cornerstone of the World Wide Web and play a crucial role in how people find information, communicate, and conduct business online [2].

3 Types of websites

Websites can be broadly categorized into two types based on how content is served.

3.1 Static websites

Web pages are prebuilt using simple languages like HTML, CSS, or JavaScript.
Characteristics:

- No server-side processing; pages are delivered as-is.
- Fast loading due to minimal server interaction.
- No database interaction.
- Cost-effective, as they do not require advanced server-side processing.

Use Case: Ideal for simple, unchanging content, e.g. personal portfolios, company brochure sites [3].

3.2 Dynamic websites

Web pages are generated at runtime using server-side scripting languages like PHP, Node.js, or ASP.NET.
Characteristics:

- Pages are built based on user requests, enabling real-time updates.
- Slower than static websites due to server-side processing.
- Supports database interaction for dynamic content.
- Easier to update, as changes can be applied globally without altering individual pages.

Use Case: Preferred for websites requiring frequent updates or user interaction, e.g. e-commerce and news websites [3].

4 Key differences between static and dynamic websites

Static and dynamic websites serve different purposes, and each has distinct advantages and disadvantages. Think of static websites as snapshots of content that remain unchanged unless manually updated. Dynamic websites can change based on user input or real-time data, creating a more interactive and personalized experience. However, they can be more complex to develop and maintain and may experience performance issues if not optimized properly.

Feature	**Static websites**	**Dynamic websites**
Content	Remains fixed	Can change based on user input or real-time data
Development	Typically simpler	More complex, requiring server-side scripting and databases
Performance	Generally faster	Can be slower, especially with complex interactions
Security	Generally more secure	Can be less secure due to potential vulnerabilities
Cost	Often more cost-effective	Can be more expensive to develop and maintain
Use cases	Simple websites, brochures, landing pages	E-commerce stores, social media platforms, news websites

Table 1.1: Comparison between static and dynamic websites [4].

5 Operating principle of websites

5.1 Web server

A web server is a computer that continuously stores, shares, and retrieves content on the internet. Web servers are the infrastructure that enables this content distribution across browsers using Hypertext Transfer Protocol (HTTP). Depending on the client's requests, some contents may be static or dynamic files [5].

A web server consists of the following components:

- **Software:** uses HTTP to handle users' requests and responses on a website.
- **Hardware:** computers that store website files such as HTML, CSS, and JavaScript files; examples include CPU, RAM, GPU, and SSDs.
- **Protocols:** facilitate intercommunication between web servers and computers; include HTTPS, SMTP, and FTP.

A web server receives, processes, and retrieves requests and sends responses through the following steps:

- **Receiving the request:** When a web browser queries a web server for information, the server's first task is to receive the request.

- **Processing the request:** The server translates the domain name into an IP address and locates the requested resource, such as an HTML page, image, or script.
- **Retrieving the requested resource:** The HTTP server locates the needed content. For a static file, it simply reads it from the file system; otherwise it runs server-side code to generate the content.
- **Sending the response:** The server returns the data to the client browser via HTTP, or sends an error message if the resource is not found [5].

5.2 Web browser

A web browser is an application through which users can access websites on the World Wide Web. It provides a medium for interaction with a web server, fetching data and displaying it as a web page [6].

5.3 Programming languages used for web development

Programming languages used for web development are divided into two main categories depending on which side of the application they operate on.

5.3.1 Front end development languages

Front end development languages are the programming languages used to develop the part of a website a user can see. When you visit a website on your device, what you see is the front end. It is the graphical user interface of a website, and it is built with HTML, CSS, and sometimes JavaScript [7].

5.3.2 Back end development languages

If there is a front, then there has to be a back. The back end of a website is the part a user cannot see. For dynamic sites, this is where all the work gets done. When you fill out a form, the backend languages handle it. The back end is built with PHP, Python, Java, and much more [7].

5.4 Database server

A database server is a specialized server dedicated to managing and providing access to databases. It serves as a centralized repository for storing structured data and enables multiple clients or applications to interact with the database simultaneously [8].

5.5 Client-server relationship

In the client-server relationship, clients send requests to the database server for data retrieval, manipulation, or administrative tasks. The server processes client requests, executes database operations, and returns responses. It ensures data integrity, concurrency control, and scalability, allowing multiple clients to access the database concurrently [8].

6 Web architectures

6.1 1-Tier Architecture

The user works directly with the database on the same system. The client, server, and database are all in one application [9].

6.2 2-Tier Architecture

Similar to a basic client-server model. The application at the client end directly communicates with the database on the server side. APIs like ODBC and JDBC are used for this interaction [9].

6.3 3-Tier Architecture

There is an additional layer between the client and the server. The client interacts with an application server, which further communicates with the database system. This intermediate layer acts as a medium for the exchange of partially processed data. This architecture is used for large web applications [9].

6.4 N-Tier Architecture

Divides an application into logical layers and physical tiers. Each layer has a specific responsibility; a higher layer can use services in a lower layer, but not vice versa. Physical separation of tiers improves scalability and resiliency but also adds latency from extra network communication [10].

7 Conclusion

In this chapter, we explored the basics of the web, from what websites and web applications are, to how they work behind the scenes with servers, browsers, and databases. We also looked at the different development languages and architectures that make these applications possible. Understanding these concepts is essential for building efficient and user-friendly web solutions, like the hospital patient management system we aim to develop in this project.

Chapter2

DESIGN AND ANALYSIS

1 Introduction

In this chapter, we present the design and modeling of our hospitalized patients management system using UML (Unified Modeling Language). First, we introduce UML and describe the main functionalities of the system. Then, we present the different diagrams created during the analysis phase, such as use case diagrams, sequence diagrams, and class diagrams. After that, we explain the relational model and the rules used to transform the class diagram into a database schema.

2 Functional and Non-functional Requirements

In this section, we distinguish between functional requirements, which describe the services that the system must provide, and non-functional requirements, which define the quality constraints such as security, performance, and usability that the system must meet.

2.1 Functional Requirements

The functional requirements represent the expected features of the platform. They can be grouped as follows:

- **User and role management:** The system must allow user authentication (doctors, nurses, medical secretaries, administrators) and role-based access with specific permissions for each user type.
- **Patient management:** The system must allow the registration of new patients, as well as the consultation and update of their personal and medical information.
- **Medical record management:** The system must provide access to patient medical records to view history, allergies, conditions, and ongoing treatments, while avoiding file duplication or loss.
- **Admission and bed management:** The system must handle patient admission and readmission, as well as bed occupancy management by department.
- **Appointment management:** The system must allow the creation, modification, and viewing of appointments between patients and doctors.
- **Care and patient monitoring:** Nurses must be able to record vital signs and validate administered treatments.
- **Staff management:** The administrator must be able to create, modify, and delete hospital staff accounts.

2.2 Non-functional Requirements

Non-functional requirements define the quality constraints imposed on the system:

- **Security:** The system must ensure the protection of medical data through secure authentication and strict access control based on user roles.

- **Usability (Ergonomics):** The interface must be simple, intuitive, and easy to use for all hospital staff, even non-technical users.
- **Performance:** The system must respond quickly to requests and support multiple users simultaneously without slowing down.

3 UML Definition

A UML (Unified Modeling Language) diagram is a visual representation of a system that shows how different components interact. Whether designing software, mapping out business processes, or planning a project, UML diagrams help organize complex ideas into clear, structured visuals [11].

4 Detailed System Description

The hospitalized patient management web application is designed to help hospitals efficiently manage and monitor hospitalized patients. This system reduces manual administrative tasks and improves the organization of medical and patient records. Below are the key components and features:

- **User Interface:** A user-friendly web interface where administrators, doctors, and nurses can interact with the system. Medical staff will have access to patient information, hospitalization details, treatment records, room assignments, and medical reports.
- **Patient Tracking and Management:** A backend system that allows hospital staff to register patients, monitor their medical status, manage admissions and discharges, assign rooms, and track treatments and appointments.
- **Database Management:** A secure database for storing and managing patient details, doctors' information, medical records, treatments, room data, and hospitalization history. It ensures data security, integrity, and efficient access.
- **Security Measures:** Role-based authentication ensuring that administrators, doctors, and nurses can only access information related to their responsibilities.

5 Use Case Diagram

5.1 Definition

A use case diagram is a visual representation of the interactions between actors (users or external systems) and a system under consideration. It depicts the functionality or behavior of a system from the user's perspective and captures the functional requirements of a system [12].

5.2 Identification of Actors

Administrator:

- Has full access to the system and all its functionalities.

- Can create, modify, and delete staff accounts (doctors, nurses, and medical secretaries).
- Can view hospital statistics and global reports.
- Oversees the overall management of the system.

Doctor:

- Can access his personal profile and update basic information.
- Can view the list of patients in his department.
- Can access full medical files of his patients.
- Can prescribe treatments and write medical reports.
- Can view and manage his appointments with patients.

Nurse:

- Can access the list of patients assigned to his department.
- Can view patient medical files.
- Can record vital signs and update patient follow-up information.
- Can validate treatments by checking boxes after execution.

Medical Secretary:

- Responsible for administrative hospital operations.
- Can register new patients and manage their admission.
- Can discharge patients from the hospital.
- Can schedule appointments between doctors and patients.
- Can view patient files but only personal information (no medical details).
- Can monitor room occupancy and manage room assignments.

5.3 Description of Interactions

The table below describes the interactions between the actors and the system.

User	Use Case	Description
Medical Secretary	Register new patient	Store a new patient's personal and administrative information in the system.
	Schedule appointments	Organize appointments between patients and doctors.
	Edit patient information	Modify patient details and hospitalization information.
	Discharge patient	Update the patient status after completing discharge procedures.

User	**Use Case**	**Description**
	View appointments	Access the list of planned patient appointments.
Doctor	View patient file	Consult the patient's medical history and hospitalization data.
	Prescribe treatment	Define medications and treatment plans for patients.
	Write evaluation	Record medical evaluations and patient progress notes.
Nurse	View patient file	Access patient information needed for daily care.
	Execute treatment schema	Apply treatments according to the doctor's instructions.
	Record vital signs	Record temperature, blood pressure, heart rate, and other vital signs.
Administrator	Manage staff account	Handle staff accounts and user permissions in the system.
	Create account	Add new accounts for hospital staff members.
	Delete account	Remove staff accounts from the system.
	View hospital statistics	Access reports and statistics related to hospital activities.

Table 2.2: Actor interactions with the system

5.4 General Use Case Diagram

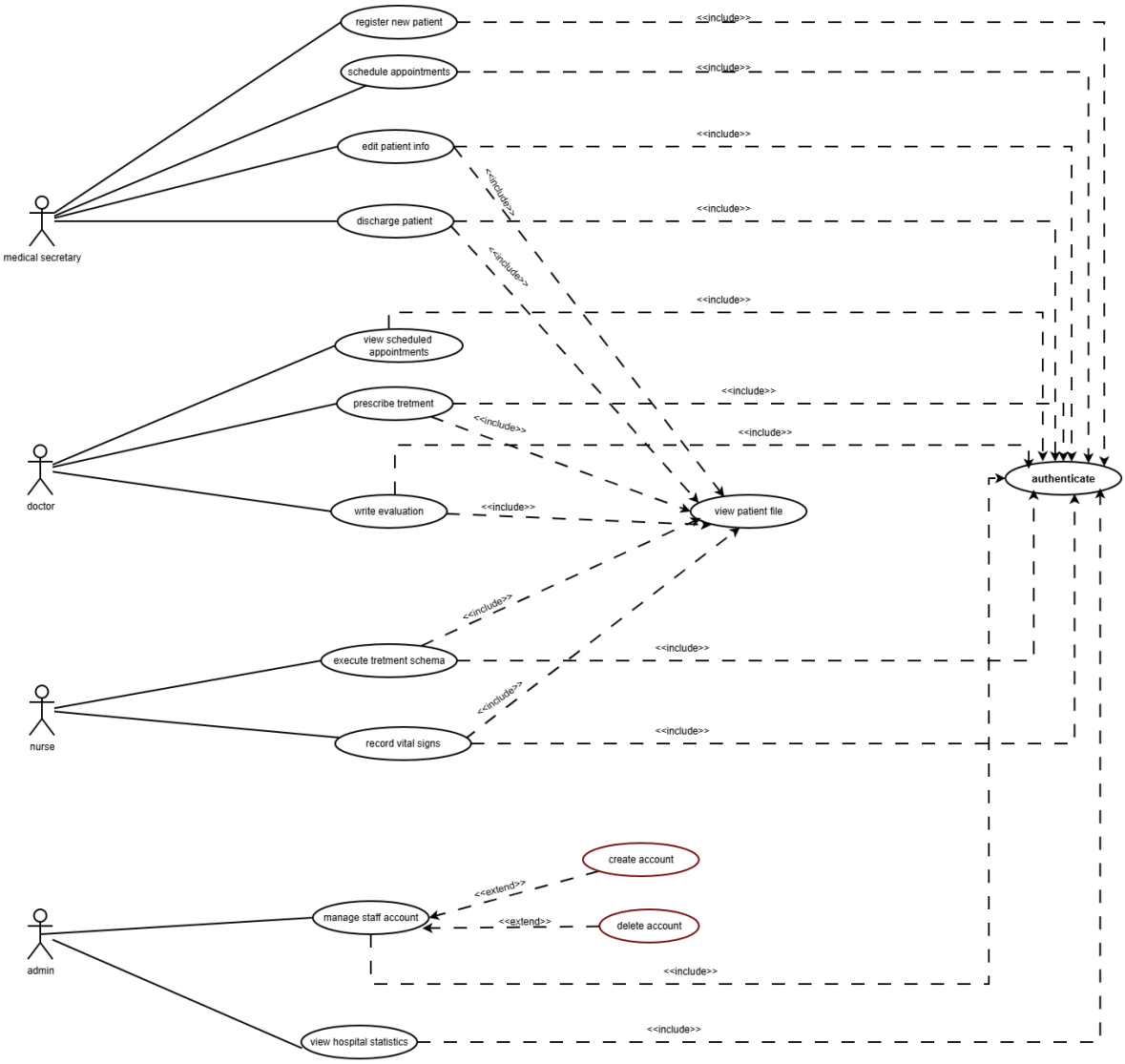


Figure 2.1: General Use Case Diagram

6 Sequence Diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time-focused and show the order of the interactions visually on the vertical axis [13].

6.1 record vital signs Sequence Diagram

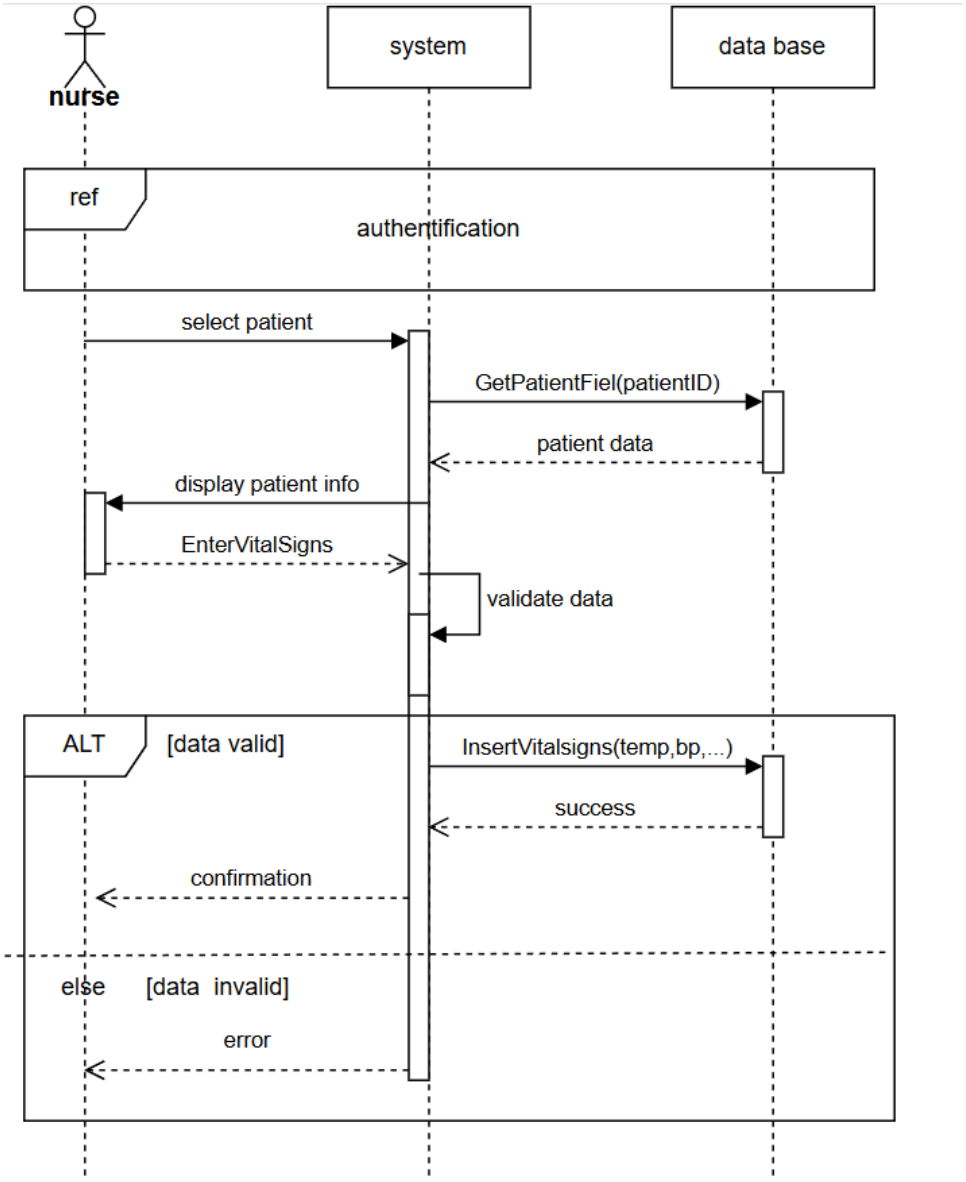


Figure 2.2: Sequence Diagram – vital signs record

6.2 creation user acount Sequence Diagram

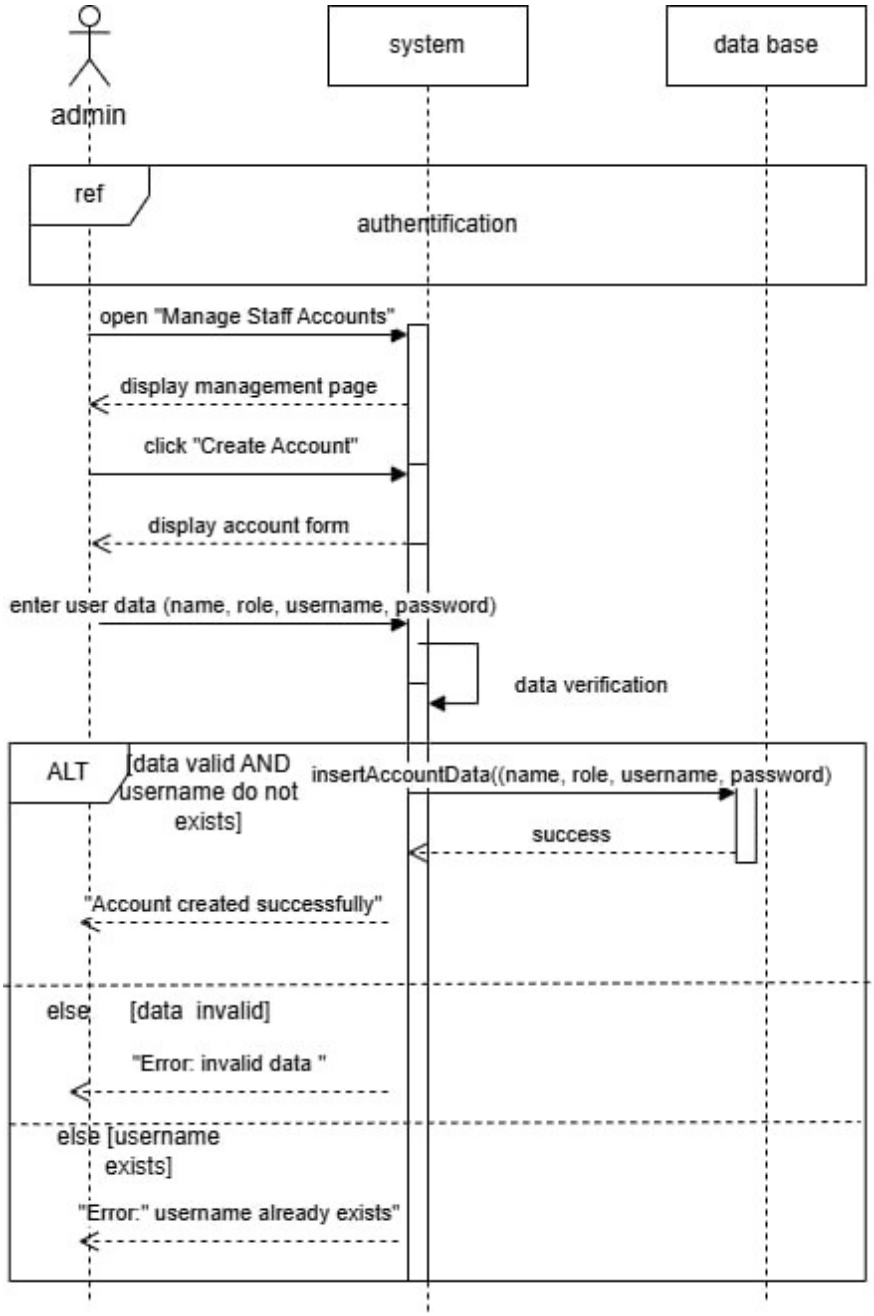


Figure 2.3: Sequence Diagram – acount creation

6.3 Appointment Scheduling Sequence Diagram

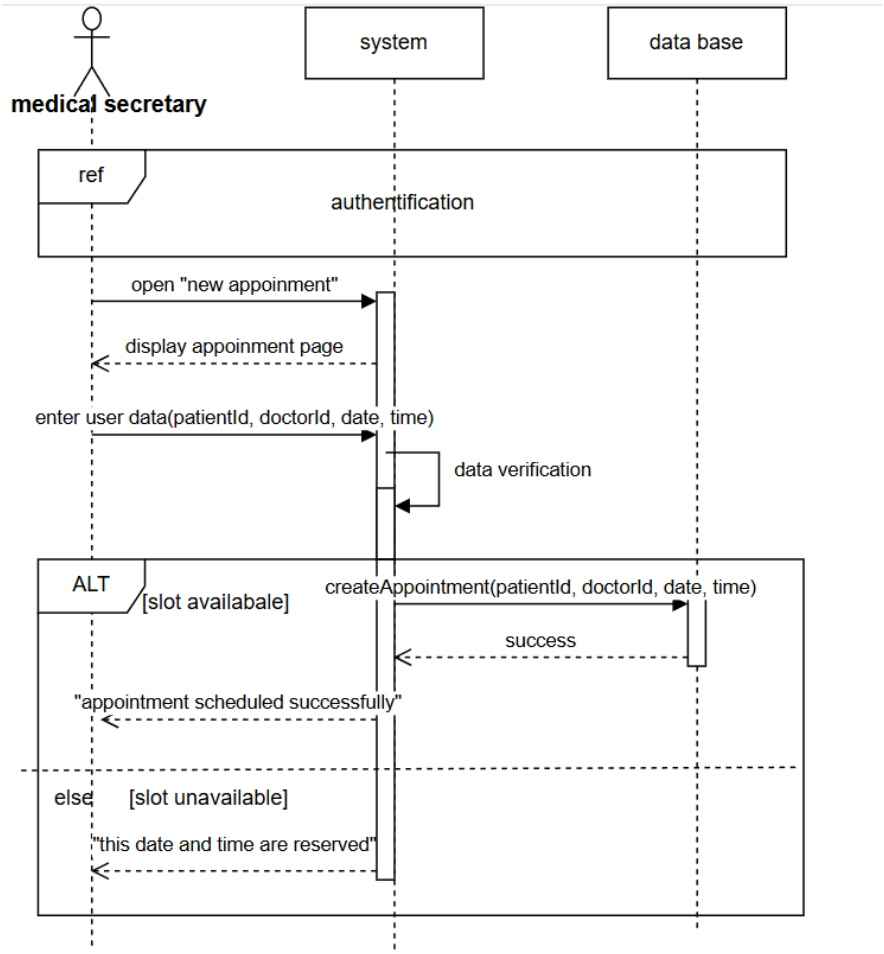


Figure 2.4: Sequence Diagram – appointment scheduling

7 Class Diagram

A class diagram is a type of UML diagram that visually represents the structure of a system by showing its classes, attributes, methods, and relationships. It serves as a blueprint for how different parts of an object-oriented system interact and organize data [14].

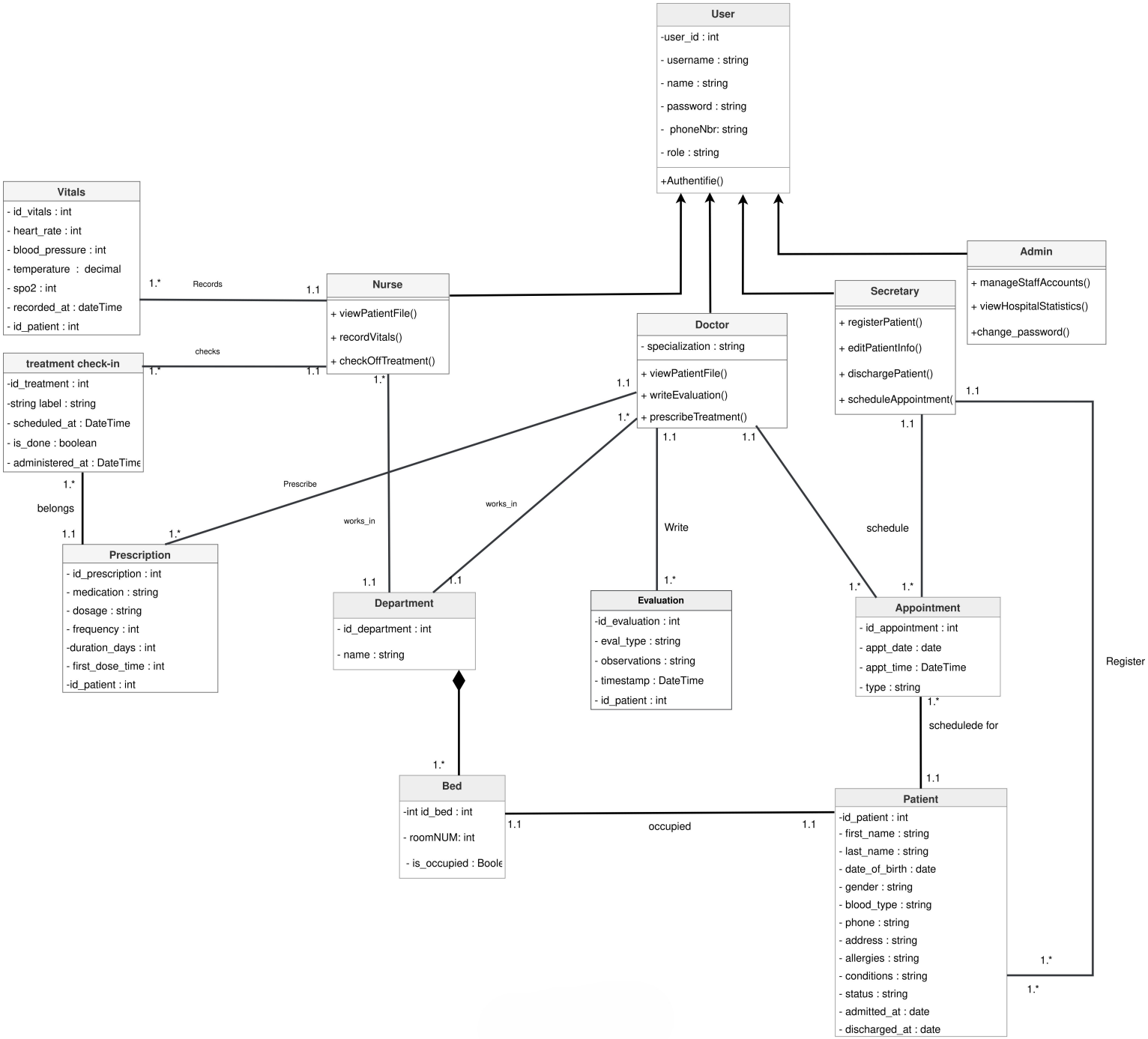


Figure 2.5: Class Diagram of the System

8 Relational Model

8.1 Definition

The relational data model provides a standard way of representing and querying data that can be used by any application. Its chief strength lies in the use of tables, which are an intuitive, efficient, and flexible way to store and access structured information [15].

8.2 Rules for Converting a Class Diagram

Rule 1 – Mapping Classes. Each class is converted into a table. Class attributes become table columns. The class identifier (ID) becomes the primary key.

Rule 2 – Mapping Inheritance. Create a separate table for each subclass. The parent table's primary key is included as a foreign key in each subclass table. Common attributes remain in the parent table, while specific attributes are stored in subclass tables.

Rule 3 – Mapping a 1-N Relationship. Add the primary key of the "one" side as a foreign key in the "many" side table.

Rule 4 – Mapping an M-N Relationship. Create a junction table between the two related tables. Include the primary keys of both tables as foreign keys. Add relationship attributes if needed.

8.3 Relational Schema

Admin (id_admin, name, username, password, phoneNbr, role)

Secretary (id_secretary, name, username, password, phoneNbr, role)

Doctor (id_doctor, name, username, password, phoneNbr, role, specialization, #id_department)

Nurse (id_nurse, name, username, password, phoneNbr, role, #id_department)

Department (id_department, name)

Bed (id_bed, roomNUM, is_occupied, #id_department)

Patient (id_patient, first_name, last_name, date_of_birth, gender, blood_type, phone, address, allergies, conditions, status, admitted_at, discharged_at, #id_bed, #id_secretary)

Appointment (id_appointment, appt_date, appt_time, type, #id_patient, #id_doctor, #id_secretary)

Evaluation (id_evaluation, eval_type, patient_state, observations, timestamp, #id_patient, #id_doctor)

Prescription (id_prescription, medication, dosage, frequency, duration_days, first_dose_time, #id_patient, #id_doctor)

Treatment_CheckIn (id_treatment, label, scheduled_at, is_done, administered_at, #id_prescription, #id_nurse)

Vitals (id_vitals, heart_rate, blood_pressure, temperature, spo2, recorded_at, #id_patient, #id_nurse)

9 Conclusion

This chapter presented the design and analysis of the hospitalized patients management system using UML diagrams and the relational model. The use case, sequence, and class diagrams helped describe the system structure, user interactions, and main functionalities. The relational model was then created to prepare the database structure for implementation. This analysis phase provides a clear foundation for the development of the application.

Chapter3

SYSTEM IMPLEMENTATION

3.1 Introduction

In this chapter, we focus on the concrete realization of our project through its implementation. We present the tools and technologies used to develop our web application, and illustrate the different interfaces through a series of screenshots highlighting the features we have designed.

3.2 Programming languages

3.2.1 HTML

HTML, or HyperText Markup Language, is a markup language for the web that defines the structure of web pages. It is one of the most basic building blocks of every website [16].



Figure 3.1: HTML5 logo [17]

3.2.2 CSS

CSS (Cascading Style Sheets) is the language used to style a web page. It describes how HTML elements are to be displayed on screen, paper, or in other media. It saves a lot of work by controlling the layout of multiple web pages all at once [18].



Figure 3.2: CSS3 logo [19]

3.2.3 JavaScript

JavaScript is the programming language of the web. It can calculate, manipulate, and validate data. It can update and change both HTML and CSS dynamically [20].



Figure 3.3: JavaScript logo [21]

3.2.4 Node.js

Node.js is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command-line tools, and scripts [22].



Figure 3.4: Node.js logo [23]

3.2.5 Express.js

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is a lightweight and flexible routing framework with minimal core features meant to be augmented through the use of Express middleware modules [24].



Figure 3.5: Express.js logo [25]

3.3 Development tools

3.3.1 MySQL

The SQL part of "MySQL" stands for "Structured Query Language." SQL is the most common standardized language used to access databases. MySQL is the most popular open-source SQL database management system, developed, distributed, and supported by Oracle Corporation [26].



Figure 3.6: MySQL logo [27]

3.3.2 Git

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency [29].



Figure 3.7: Git logo [28]

3.3.3 GitHub

GitHub is a cloud-based platform where you can store, share, and work together with others to write code, and track and manage changes to your code over time [30].



Figure 3.8: GitHub logo [31]

3.3.4 Visual Studio Code

Visual Studio Code is a powerful integrated development environment (IDE) where you can develop, build, debug, test, and deploy apps, all in one place. It includes compilers, code completion tools, source control, extensions, and many more features to enhance every stage of the software development process [32].



Figure 3.9: Visual Studio Code logo [33]

3.3.5 Draw.io

Draw.io is a technology stack for building diagramming applications, and the world's most widely used browser-based end-user diagramming software [34].



Figure 3.10: Draw.io logo [35]

3.4 Presentation of the application's interfaces

In this section, we present some interfaces of our platform.

3.4.1 Create Stuff Account Page

The figure below shows the interface that displays the Creation Account form.

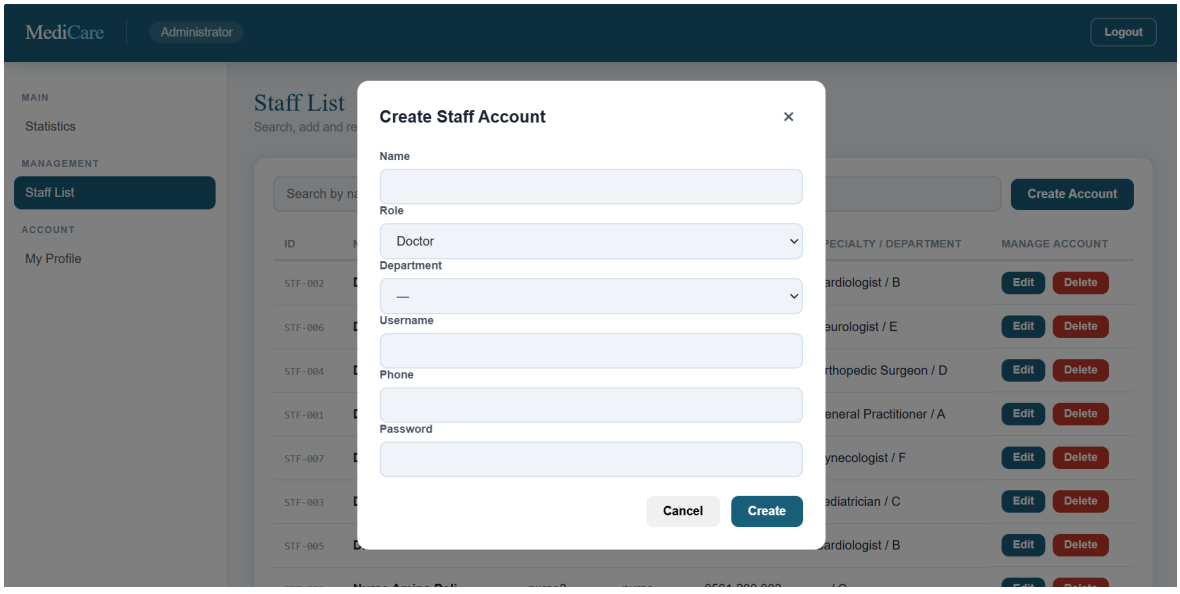


Figure 3.11: Create Account page

3.4.2 Medical Secretary Dashboard

The figure below shows the interface that displays the Dashboard page.

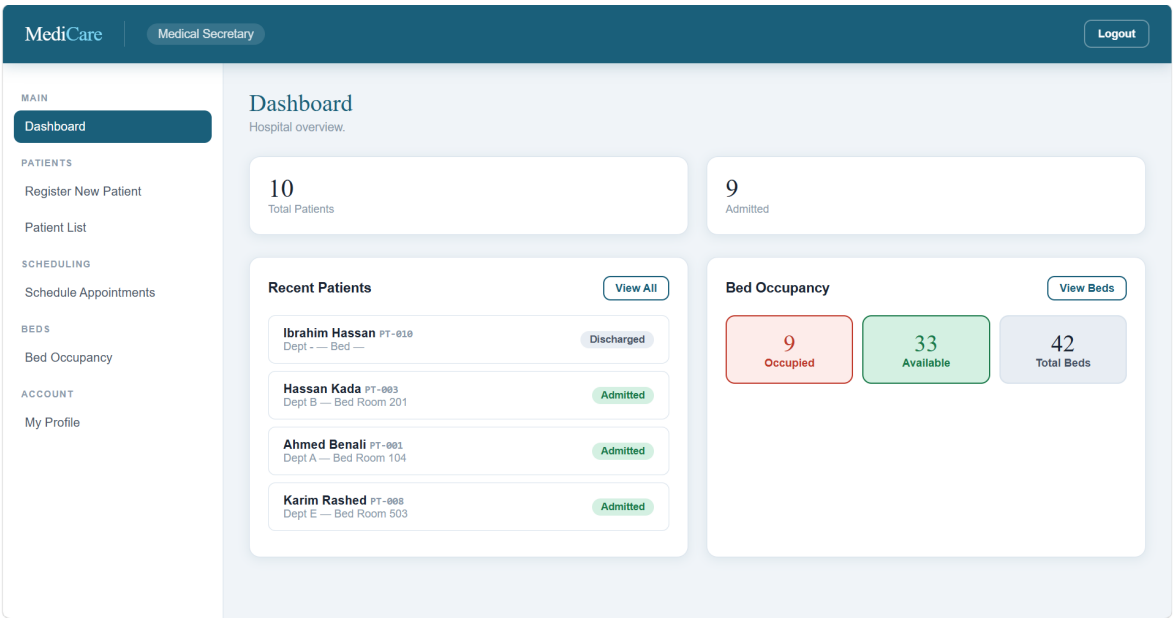


Figure 3.12: Schedule Appointment Page

3.4.3 Registration Page

The figure below shows the interface that displays the registration of new patient form.

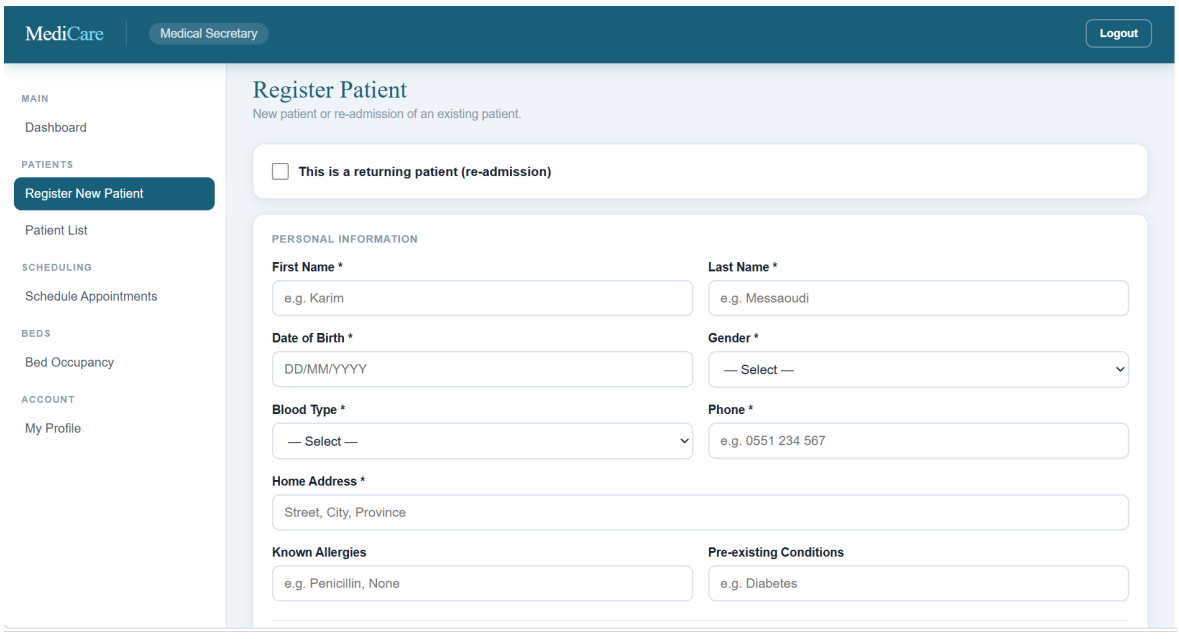


Figure 3.13: register new patient page

3.4.4 Schedule Appointment Page

The figure below shows the interface that displays the Schedule Appointment page.

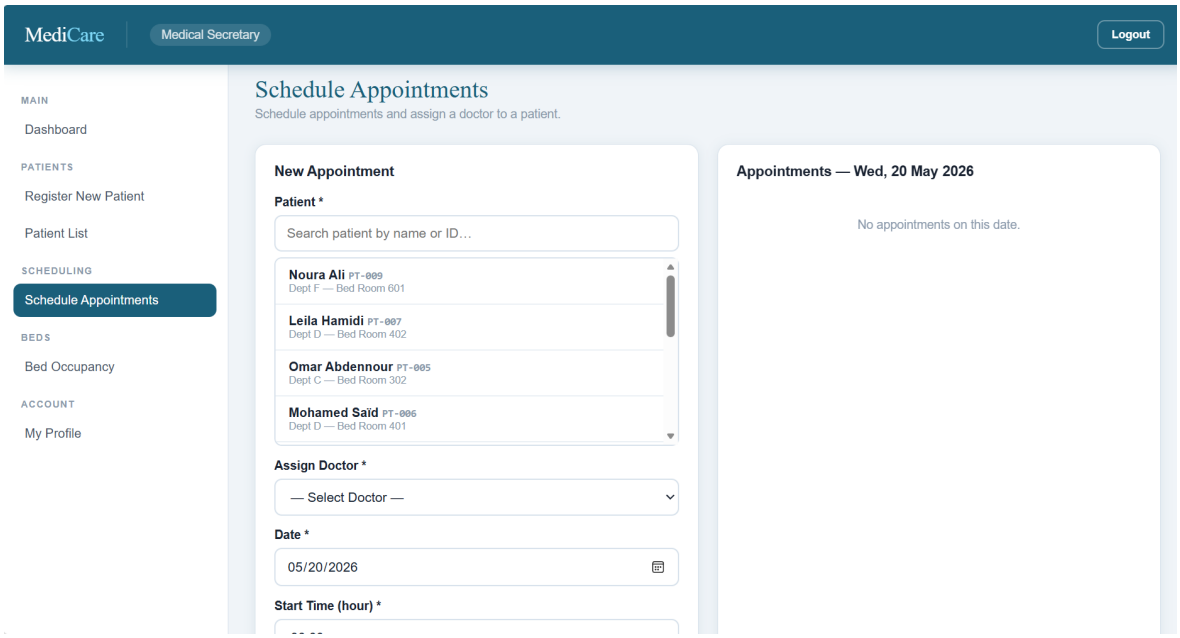


Figure 3.14: Schedule Appointment Page

3.5 Conclusion

In this chapter, we presented the programming languages and development tools used, as well as the main pages of the website. This gives a clear overview of how the project was developed and structured.

General Conclusion

At the end of this work devoted to the design and implementation of a web application to manage hospitalized patients, it is appropriate to take stock of the work accomplished and measure the achievement of the objectives set.

Our project was based on a simple observation: hospital staff had to face increasing complexity in managing patient data, without a centralized tool to manage patient care efficiently from admission to discharge. This lack of a common platform resulted in information silos, loss of files, and problems in the coordination of medical activities between departments.

The project is fairly successful. We successfully created *MediCare*, a working web platform that meets the set objectives. The system now allows central management of patients and staff, a full hospitalization cycle, access to medical records, appointment scheduling, and bed occupancy management, all built on modern technologies such as Node.js and Express.js.

This project was an enriching experience that allowed us to connect theory and practice, apply good software engineering principles, and develop skills in project management and real-world problem solving. It represents a solid foundation that can be further improved and extended to better support hospital management needs in the future.

Bibliography

[1] Adobe Systems, *What is a web application?*, https://helpx.adobe.com/dreamweaver/using/web-applications.html

[2] OVHcloud, *What is a website?*, https://www.ovhcloud.com/en/learn/what-is-website/

[3] GeeksForGeeks, *What is a Website?*, https://www.geeksforgeeks.org/blogs/what-is-a-website/

[4] Figma, *Static vs Dynamic Website*, https://www.figma.com/resource-library/static-vs-dynamic-website/

[5] Liquid Web, *What is a Web Server?*, https://www.liquidweb.com/blog/what-is-a-web-server/

[6] GeeksForGeeks, *Functions and Features of a Web Browser*, https://www.geeksforgeeks.org/websites-apps/functions-and-features-of-a-web-browser/

[7] Loop Studio, *Best Programming Languages in Web Development*, https://loopstudio.dev/best-programming-languages-in-web-development/

[8] Lenovo, *What is a Database Server?*, https://www.lenovo.com/ca/en/glossary/database-server/

[9] GeeksForGeeks, *DBMS Architecture – 2-Level & 3-Level*, https://www.geeksforgeeks.org/dbms/dbms-architecture-2-level-3-level/

[10] Microsoft Azure, *N-tier architecture style*, https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier

[11] Atlassian, *What Is a UML Diagram*, https://www.atlassian.com/software/confluence/templates/uml-diagram

[12] Creately, *Use Case Diagram Tutorial*, https://creately.com/guides/use-case-diagram-tutorial/

[13] Visual Paradigm, *What is a Sequence Diagram?*, https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

[14] Creately, *What is a Class Diagram*, https://creately.com/guides/class-diagram-tutorial/

[15] Oracle, *What Is a Relational Database?*, https://www.oracle.com/database/what-is-a-relational-database/

[16] freeCodeCamp, *What is HTML?*, https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/

[17] W3Schools, *What is HTML?*, https://www.w3schools.com/whatis/whatis_html.asp

[18] W3Schools, *CSS Introduction*, https://www.w3schools.com/css/css_intro.asp

[19] W3Schools, *What is CSS?*, https://www.w3schools.com/whatis/whatis_css.asp

[20] W3Schools, *JavaScript Introduction*, https://www.w3schools.com/js/js_intro.asp

[21] W3Schools, *What is JavaScript?*, https://www.w3schools.com/whatis/whatis_js.asp

[22] Node.js Foundation, *About Node.js*, https://nodejs.org/en

[23] Node.js Foundation, *Node.js Branding*, https://nodejs.org/en/about/branding

[24] OpenJS Foundation, *Express – Node.js web application framework*, https://expressjs.com/

[25] Wikimedia Commons, *Express.js logo*, https://commons.wikimedia.org/wiki/File:Expressjs.png

[26] Oracle Corporation, *MySQL 8.4 Reference Manual*, https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html

[27] Oracle Corporation, *MySQL Logos*, https://www.mysql.com/about/legal/logos.html

[28] Git Project, *Git – fast, scalable, distributed revision control system*, https://git-scm.com/

[29] Git Project, *Git Logos*, https://git-scm.com/community/logos

[30] GitHub, *About GitHub and Git*, https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

[31] GitHub, *GitHub Logo*, https://brand.github.com/foundations/logo

[32] Microsoft, *Visual Studio IDE overview*, https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide

[33] GitHub, *Logo Designing*, https://github.com/topics/logo-designing

[34] Wikimedia Commons, *Draw.io Logo*, https://commons.wikimedia.org/wiki/File:Diagrams.net_Logo.svg

[35] JGraph Ltd., *About draw.io*, https://www.drawio.com/about