

# Web-based online medicine service system



A Project presented to the National University in partial fulfillment of the requirement for the degree of Bachelor of Science (Hon's) in Computer Science & Engineering.

## Supervised By:

**Md. Imran Hossain**

Head

Department of Computer Science & Engineering  
Daffodil Institute of IT

## Submitted By:

**Moynul Islam**

Registration No: 175002004917

**Session: 2017-18**



Daffodil Institute of IT

Department of Computer Science and Engineering

Daffodil Institute of IT,

Under National University (NU)- Dhaka, Bangladesh

Submission Date:04/09/2023

## **APPROVAL**

This Project titled “**Web-based online medicine service system**” Is submitted to the Department of Computer Science & Engineering (CSE) of Daffodil Institute of IT (DIIT) Under National University (NU). It has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor in Computer Science & Engineering (CSE) & approved as to its styles & contents.

---

EXAMINAR

---

EXAMINAR

---

Md. Imran Hossain  
Head & Project Supervisor  
Department of Computer Science & Engineering  
Daffodil Institute of IT

## **DECLARATION**

I hereby declare that the project worked entitled “**Web-based online medicine service system**” submitted to the degree of BSc. (Hons) in Computer Science & Engineering (CSE) is a record of original work done by me. Except as acknowledged in the text and that the material has not been submitted, either in whole or in part, for a degree at this or any other university.

**Submitted by:**

-----

Moynul Islam

Registration No: 175002004917

Session:2017-18

## ACKNOWLEDGMENT

Despite our efforts, the success of this project depends largely on the encouragement and guidance of our mentors. We would like to take this opportunity to express our gratitude to the people who are playing a vital role in the successful completion of this project.

My sincere thanks to the **Prof. Dr. Mohammed Shakhawat Hossain**, Principle, DIIT who has allowed us to work on this project and showed encouragement.

My cordial thanks to our Project Supervisor **Md. Imran Hossain**, Head, Department of Science & Engineering, DIIT for her valuable guidance and support to meet the successful completion of our project.

My heartiest thanks to **Saidur Rahman**, Senior Lecturer, Department of Computer Science & Engineering, DIIT for his patronage and giving us an opportunity to undertake this Project.

I express my gratitude to **Poly Bhoumik**, Senior Lecturer, Department of Computer Science & Engineering, DIIT for providing us the facilities to do the project successfully.

I also say thanks to **Safrun Nesa Saira**, Lecturer, Department of Computer Science & Engineering, DIIT for providing us the facilities to do the project successfully.

I express my gratitude to **Mizanur Rahman**, Lecturer, Department of Computer Science & Engineering, DIIT for providing us the facilities to do the project successfully.

I also express my gratitude to **Nusrhat Jahan Sarker**, Lecturer, Department of Computer Science & Engineering, DIIT for providing us the facilities to do the project successfully.

We extend our sincere thanks to our family & classmates for their constant support throughout this project.

Finally, we would be grateful to the **National University, Bangladesh** and the coordinators of the **Bachelor of Science in Computer Science and Engineering** degree program for giving us this opportunity to apply the knowledge that we have gained through the study of the degree program.

## **ABSTRACT**

The project "Web-based online medicine service system" aims to computerize and enhance the e-commerce system for medicine sales, prioritizing user-friendliness, speed, and cost-effectiveness. It seeks to improve accuracy, safety, and efficiency in pharmacy management through a web-based platform. The system employs MySQL for database management, PHP Laravel for the backend, and follows a 3-tier architecture. It offers streamlined operations, strict cost control, and improved profitability while receiving positive feedback and support.

# TABLE OF CONTENTS

Approval .....	i
Declaration .....	ii
Acknowledgments.....	iii
Abstract .....	iv
Table of Contents.....	v-viii
<b>Chapter 1: Introduction .....</b>	<b>1-4</b>
1.1 Introduction.....	1
1.2 Problem introduction... ..	3
1.2.1 Lack of immediate retrievals... ..	3
1.2.2 Lack of immediate information storage.....	3
1.2.3 Lack of prompt updating.....	3
1.2.4 Error-prone manual calculation... ..	3
1.2.5 Preparation of accurate and prompt reports... ..	3
1.3 Objectives .....	3
1.4 Scope of the project... ..	4
<b>Chapter 2: Background Study.....</b>	<b>5-6</b>
2.1 Analysis of existing system .....	6
2.2 Software system attribute.....	6
2.2.1 Reliability.....	6
2.2.2 Availability .....	6
2.2.3 Security .....	6
<b>Chapter 3: System Analysis .....</b>	<b>7-13</b>

3.1 SDLC .....	8
3.1.1 Waterfall model .....	8
3.1.2 Incremental model .....	8
3.1.3 Iterative model .....	8
3.1.4 Sprital model.....	9
3.1.5 V-Shape model .....	9
3.1.6 Agile method.....	9
3.2 Necessity of SDLC.....	9
3.2.1 SDLC for our project. . . . .	10
3.3 Agile method.....	10
3.3.1 Advantages of agile method.....	11
3.3.2 Disadvantages of the agile method. . . . .	11
3.3.3 When to use the agile method. . . . .	11
3.4 Study of the system .....	11
3.4.1 Existing system. . . . .	11
3.4.2 Proposed system. . . . .	12
3.5 Modules.....	12
3.5.1 Admin module .....	12
3.5.2 User module .....	12
3.5.3 Doctor module .....	13
<b>Chapter 4: Feasibility Report .....</b>	<b>14-15</b>
4.1 Feasibility study .....	15
4.1.1 Economic feasibility .....	15
4.1.2 Technical feasibility.....	15
4.1.3 Operational feasibility.....	15
<b>Chapter 5: System Design .....</b>	<b>16-30</b>
5.1 Introduction to UML.....	17

5.1.1 UML design .....	17
5.1.2 Visualizing .....	17
5.1.3 Specifying .....	17
5.1.4 Constructing... ..	17
5.1.5 Documenting... ..	18
5.2 UML approach... ..	18
5.2.1 UML diagram... ..	18
5.2.2 Use Case Diagram.....	18
5.2.3 Flow Chart... ..	20-23
5.3 Data Flow Diagram (DFD).....	24
5.3.1 Data flow diagram for patient register .....	25
5.3.2 Level 0 .....	25
5.3.3 Level 1... ..	25
5.3.4 Level 2... ..	26
5.4 ER diagram... ..	26-28
5.5 Activity diagram... ..	28-30
<b>Chapter 6: Requirement Specification.....</b>	<b>31-37</b>
6.1 Introduction.....	32
6.2 Hardware requirements... ..	32
6.2.1 Hardware requirement for present project .....	32
6.3 Software requirements... ..	32
6.3.1 Software requirement for present project... ..	33
6.4 Software specification... ..	33
6.4.1 HTML .....	33
6.4.2 Cascading style sheets (CSS).....	34
6.4.3 MySQL .....	34
6.4.4 Features of MySQL.....	35
6.4.5 Security .....	35
6.4.6 Scalability and limits... ..	35



6.4.7 Connectivity .....	35
6.4.8 Localization... ..	36
6.4.9 Clients and tools.....	36
6.4.10 Why to use MYSQL .....	36
6.4.11 JavaScript.....	36
6.4.12 Why to use JavaScript.....	37
6.4.13 Other uses of JavaScript.....	37
6.4.14 PHP.....	37
6.4.15 Why PHP.....	37
<b>Chapter 7: Code Implementation .....</b>	<b>38-47</b>
7.1 Introduction .....	39
7.2 Website Frontend .....	39
7.3 Website Backend .....	43
7.4 Code Implementation .....	44
<b>Chapter 8: Conclusion.....</b>	<b>48-50</b>
8.1 Conclusion.....	49
8.2 Limitations of the system.....	49
8.3 Future enhancement.....	49
<b>Reference.....</b>	<b>50</b>

CHAPTER 1  
**INTRODUCTION**

## **1.1 Introduction:**

The project Web-based online medicine service system, which is based on medicine and its relative works, includes registration of customers, storing their details in the system, and also computerized billing in the pharmacy and labs. The software can give a unique id for every customer and automatically stores the details of every customer and the medicine store staff. It includes a search facility to know the current status of each room. Users can search for the availability of medicine and the details of medicine using the id. The medicine service system can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data to the database. The data can be retrieved easily. The interface is very user-friendly. The data is well protected for personal use and makes the data processing very fast.

The Web-based online medicine service system is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to medicine administrators and users.

Web-based online medicine service system is designed for multispecialty medicine to cover a wide range of mass medicine and management processes. It is an integrated end-to-end medicine Management System that provides relevant information to support effective decision-making for patient care, hospital administration, and critical financial accounting in a seamless flow.

Web-based online medicine service system is a software product suite designed to improve medicine management's quality and management in medicine process analysis and activity-based costing. Web-based online medicine service system enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of medicine helps you manage your processes.

## **1.2 Problem Introduction:**

### **1.2.1 Lack of immediate retrievals:**

The information is very difficult to retrieve, and to find certain information like- E.g., To find out about the medicine history, the user has to go through various registers. This results in convenience and wastage of time.

**1.2.2 Lack of primary information storage:** The information generated by various transactions takes time and effort to be stored in the right place.

### **1.2.3 Lack of prompt updating:**

Various changes to information like patient details or immunization details of a child are difficult to make as paperwork is involved.

### **1.2.4 Error-prone manual calculation:**

Manual calculations are error-prone and take a lot of time. This may result in incorrect information. For example, the calculation of patient bills based on various treatments.

### **1.2.5 Preparation of accurate and prompt reports:**

This becomes difficult as information is difficult to collect from various registers.

## **1.3 Objective:**

Medicine stores currently use a manual system to manage and maintain critical information. The current system requires numerous paper forms, with data stores spread throughout the hospital management infrastructure. Often information (on forms) is incomplete or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost. Multiple copies of the same information exist in the medicine and may lead to inconsistencies in data in various data stores.

A significant part of the operation of any medicine store involves the acquisition, management, and timely retrieval of great volumes of information.

This information typically involves patient personal information and medicine history, staff information, medicine information bank, and various facilities.

All of this information must be managed in an efficient and cost-wise fashion so that institutions' resources may be effectively utilized. Medicine stores will automate the management of the medicine stores making it more efficient and error-free. It aims at standardizing data, consolidating data, ensuring data integrating, and reducing inconsistencies.

These are the various jobs needed in a medicine store by the operational staff and pharmacist. All these works are done on paper.

#### **1.4 Scope of the Project:**

Information about Patients is done by just writing the Patient's name, age, gender, and medicine information. Whenever the Patient comes up, his information is stored freshly. Bills are generated by recording the price for each facility provided to the Patient on a separate sheet, and at last, they all are summed up. Diagnosis information to patients is generally recorded on the document containing Patient information.

It is destroyed after some time period to decrease the paper load in the Office. Children's immunization records are maintained in pre-formatted sheets, which are kept in a file. Information about various diseases is not kept as any document. Pharmacists themselves do this job by remembering various medicines. All this work is done manually by the receptionist and other operational staff, and a lot of papers must be handled and taken care of. Pharmacists have to remember various medicines available for diagnosis and sometimes miss better alternatives as they can't remember them then.

CHAPTER 2  
**BACKGROUND STUDY**

## **2.1 Analysis of existing system:**

System Analysis is a separation of a substance into parts for study and their implementation and detailed examination.

Before designing any system, the nature of the business and the way it currently operates must be clearly understood. The detailed examination provides the specific data required during design to ensure that all the client's requirements are fulfilled. The investigation or the study conducted during the analysis phase is largely based on the feasibility study. Rather, it is not wrong to say that the analysis and feasibility phases overlap. The high-level analysis begins during the feasibility study. Though analysis is represented as one phase of the system development life cycle (SDLC), this is untrue. The analysis begins with system initialization and continues until its maintenance. Even after the successful implementation of the system, the analysis may play its role in the periodic maintenance and gradation of the system. One of the main causes of project failures is inadequate understanding, and one of the main causes of inadequate understanding of the requirements is poor planning of system analysis.

## **2.2 Software system attributes**

**2.2.1 Reliability:** This application is a reliable product that produces fast & verified output of all its process.

**2.2.2 Availability:** This application will be available to use and help them to carry out their operations conveniently.

**2.2.3 Security:** This application will be designed in a maintainable manner. It will be easy to incorporate new requirements in the individual modules.

CHAPTER 3  
**SYSTEM ANALYSIS**



### **3.1 SDLC**

While all software projects have to be professionally manned and developed, different techniques are appropriate for different types of systems. For example, games should always be developed using a series of prototypes, whereas safety-critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.

One of the basic notions of the software development process in SDLC models stands for Software Development Life Cycle models. The most used, popular, and important SDLC models are

#### **3.1.1 Waterfall model:**

The waterfall model is a classical model used in the system development life cycle to create a system with a linear and sequential approach. It is termed a waterfall because the model develops systematically from one phase to another downward. This model is divided into different phases, and the output of one phase is used as the input of the next phase. Every phase has to be completed before the next phase starts, and there is no overlapping of the phases.

#### **3.1.2 Incremental Model:**

Incremental Model is a process of software development where requirements are divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation, and testing phases. Every subsequent release of the module adds a function to the previous release. The process continues until the complete system is achieved.

#### **3.1.3 Iterative Model:**

The iterative model is a particular implementation of a software development life cycle (SDLC) that focuses on an initial, simplified implementation, progressively gaining more complexity and a broader feature set until the final system is complete.

#### **3.1.4 Spiral Model:**

The spiral model is a systems development lifecycle (SDLC) method used for risk management that combines the iterative development process model with elements of the Waterfall model. The spiral model is used by software engineers and is favored for large, expensive, and complicated projects.

#### **3.1.5 V-Shaped Model:**

The V-model is a type of SDLC model where the process executes in a sequential manner in a V-shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step is directly associated with the testing phase.

#### **3.1.6 Agile Model:**

Agile Methodology meaning a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.

For this particular project, I've used Agile Method as it is more user-friendly and easy to handle.

### **3.2 Necessity of SDLC**

Software developers follow some methodology which is a step-wise process to develop a project involving the usage of programming language for a particular solution. It includes examining, designing, & developing, testing, and documenting. Implementing and evaluating the complex subject of software engineering. Developers always try to reduce the risks.

The longer a project runs, the more risk remains of getting bugs. That is why we must follow some procedures that make our work easier and less risky. It's known as SDLC, and that's why it is necessary.

### 3.2.1 Software Development Life Cycle for Our Project

There are several SDLC methods renowned for normal development. As a project can't be completed in one phase, rather, we need to develop it primarily, and then it'll gradually learn. We'll keep developing it to make it better; we're going to use the Agile Method.

## 3.3 Agile Method

Agile software development describes a set of principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams. It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement and encourages rapid and flexible responses to change. These principles support the definition and continuing evolution of many software development methods.

If we relate to the term 'agile,' we can easily understand where it should be used. It can be implemented in all types of projects, but its essence can only be extracted if we use it on bigger and more complex projects. We can use agile when we can implement it for the success of the project, or the nature of the project requires it. Agile should not be used just as a run away from a waterfall.

We can use agile where collaboration is highly important for the success of the project, where we have long-term goals and no bound on the requirements. Agile can be well utilized where we have great power of either utilizing time as much as we want or resources as much as we want.



1.1.1.1 Fig 3.3: Agile Method Software model

### **3.3.1 Advantages of Agile Method**

The Agile methodology was firstly developed for the software industry. The task was to optimize and improve the development process and to try to identify and quickly correct problems and defects. This methodology allows for providing better output more quickly through short and interactive sessions/sprints.

In the era of digital transformation, where many organizations are migrating to a digital workplace, the agile methodology suits perfectly in companies looking to transform how projects are managed and how they operate as a whole.

### **3.3.2 Disadvantages of Agile Method**

The disadvantages of Agile methodology make clear that it's not for everyone. Mitigating the disadvantages of agile methodology requires taking more of a Lean approach by emphasizing. Delivering value through a quality end product rather than simply delivering a working product managing a clear process for delivering that product, not an uncertain route determined along the way. Needs special skills for the team, and documentation is done at later stages.

### **3.3.3 When to Use the Agile Method**

Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for time-critical applications. Extreme Programming is one of the most well-known agile development life cycle models.

## **3.4 Study of the system:**

### **3.4.1 Existing System:**

Pharmacies currently use a manual system to manage and maintain critical information. The current system requires numerous paper forms, with data

stores spread throughout the pharmacy management infrastructure. Often information is incomplete or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost. Multiple copies of the same information exist in the hospital and may lead to inconsistencies in data in various data stores

#### **3.4.2 Proposed System:**

The Web-based online medicine service system is designed for any pharmacy to replace their existing manual paper-based system. The new system is to control the information of patients. Medicine information, staff and operating room schedules, and patient invoices. These services are to be provided in an efficient, cost-effective manner, with the goal of reducing the time and resources currently required for such tasks.

### **3.5 Modules:**

The entire project mainly consists of 3 modules, which are:

- 3.5.1 the Admin module
- 3.5.2 User module (patient)
- 3.5.3 Doctor modules

#### **3.5.1 Admin module:**

- Manage department of medicine, user, and Pharmacists.
- Watch transaction reports of patient payment
- Watch medicine bank report
- Watch medicine status
- Watch diagnosis report

#### **3.5.2 User module (patient):**

- View medicines information and details
- View medicine availability and price
- View transaction details
- View payment method

### **3.5.3 Pharmacist module:**

- Manage patients. account opening and updating
- Create, manage appointment with patient
- Create prescription for patient
- Provide medication for patients

CHAPTER 4  
**Feasibility Report**

## **4.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This ensures that the proposed system is not a burden to the company. For feasibility analysis, understanding the system's major requirements is essential.

Three key considerations involved in the feasibility analysis are:

### **4.1.1 Economic Feasibility**

This study is carried out to check the economic impact the system will have on the organization. The amount of funds the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget, and this was achieved because most of the technologies used are freely available. Only customized products have to be purchased.

### **4.1.2 Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the system's technical requirements. Any system developed must not have a high demand for the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for implementing this system.

### **4.1.3 Operational Feasibility**

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system; instead, they must accept it as a necessity. The level of acceptance by the users solely depends on the methods employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he can also make some constructive criticism, which is welcomed, as he is the final user of the system.



CHAPTER 5  
**SYSTEM DESIGN**

## **5.1 INTRODUCTION TO UML**

### **5.1.1 UML Design**

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the software system and its components. It is a graphical language that provides a vocabulary and a set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain, and control information about the systems.

The UML is a language for:

- Visualizing
- Specifying
- Constructing
- Documenting

### **5.1.2 Visualizing**

Through UML we see or visualize an existing system, and ultimately, we visualize how the system will be after implementation. Unless we think, we cannot implement. UML helps to visualize how the components of the system communicate and interact with each other.

### **5.1.3 Specifying**

Specifying means building models that are precise, unambiguous, and complete UML addresses the specification of all the important analysis design, and implementation decisions that must be made in developing and deploying a software system.

### **5.1.4 Constructing**

UML models can be directly connected to a variety of programming languages by mapping a model from UML to a programming language like JAVA or C++, or VB. Forward Engineering and Reverse Engineering are possible through UML.

### **5.1.5 Documenting**

The Deliverables of a project, apart from coding, are some Artifacts, which are critical in controlling, measuring, and communicating a system during its developing requirements, architecture, design, source code, project plans, tests, prototypes, releases, etc.

## **5.2 UML Approach**

### **5.2.1 UML Diagram**

- A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices and arcs. You draw a diagram to visualize a system from a different perspective, so a diagram is a projection of a system. For all but most trivial systems, a diagram represents an elided view of the elements that make up a system. The same element may appear in all diagrams, only a few diagrams, or none. In theory, a diagram may contain any combination of things and relationships. In practice, however, a few common combinations arise, consistent with the five most useful views that comprise the architecture of a software-intensive system. For this reason, the UML includes nine such diagrams:
  - Use case diagram
  - Flow Chart
  - DFD diagram
  - ER diagram
  - Activity diagram

### **5.2.2 USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. Use case diagrams are formally included in two modeling languages defined by the OMG: the unified modeling language (UML) and the systems modeling language.



5.2.2 Figure: Use case diagram

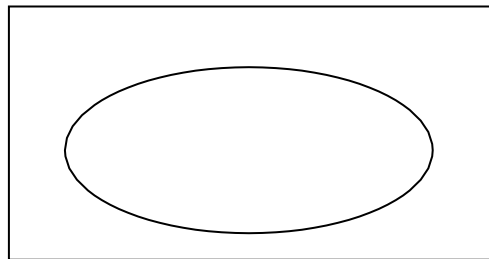
### 5.2.3 Flow Chart:

A flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols that are connected among them to indicate the flow of information and processing.

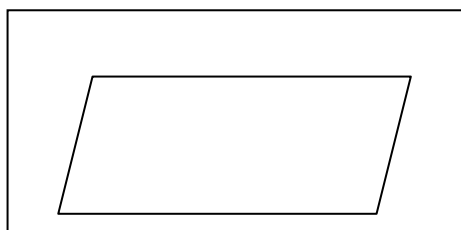
Drawing a flowchart for an algorithm is known as “flowcharting”.

Basic Symbols used in Flowchart Designs:

- **Terminal:** The oval symbol indicates Start, Stop, and Halt in a program’s logic flow. A pause/halt is generally used in a program logic under some error conditions. The terminal is the first and last symbol in the flowchart.

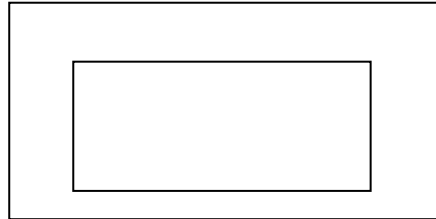


- **Input/Output:** A parallelogram denotes any input/output type function. Program instructions that take input from input devices and display output on output devices are indicated with parallelograms in a flowchart.

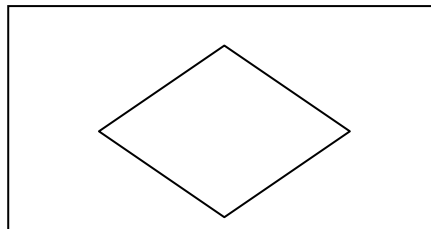


➤ **Processing:**

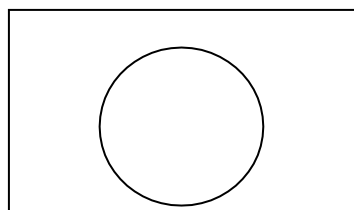
A box represents arithmetic instructions. All arithmetic processes, such as adding, subtracting, multiplication, and division, are indicated by action or process symbol.



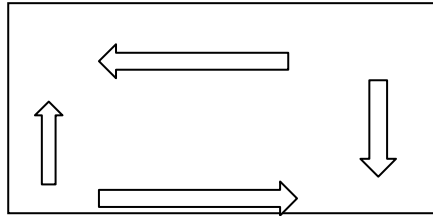
- **Decision** Diamond symbol represents a decision point. Decision-based operations such as yes/no questions or true/false are indicated by the diamond in the flowchart.

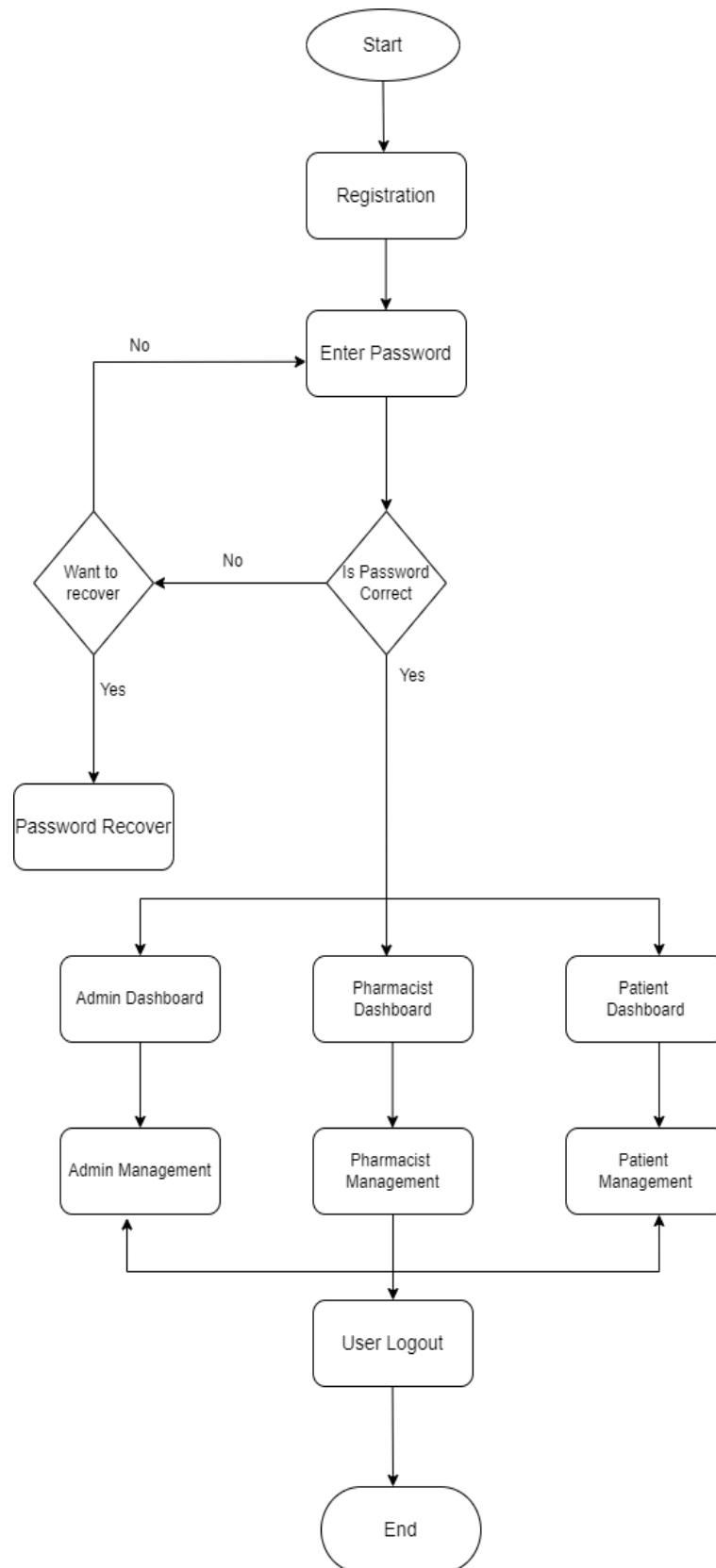


- **Connectors:** Whenever the flowchart becomes complex, or it spreads over more than one page, it is useful to use connectors to avoid any confusion. It is represented by a circle.



- **Flow lines:** Flow lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of the flow of control and the relationship among different symbols of the flowchart.





5.2.2 Figure: Flow chart



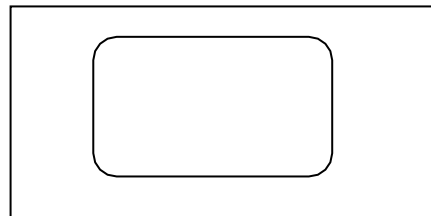
### 5.3 Data Flow Diagram (DFD)

A data flow diagram (DFD) is a diagram that describes the flow of data and the process that changes data throughout a system. It's a structured analysis and design tool used for flowcharting in place of or in association with information.

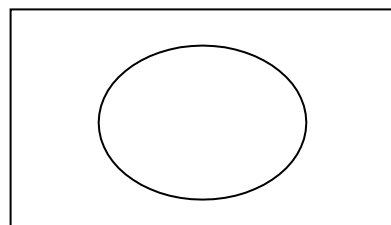
The DFD reviews the current system, prepares input and output specifications, specifies the implementation plan, etc.

Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

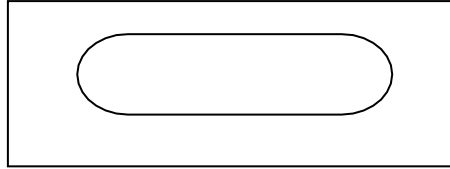
- **External Entity:** An external system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system, or a business system. They are also known as terminators, sources, and sinks or actors. They are typically drawn on the edges of the diagram.



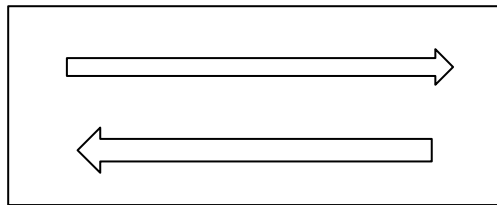
- **Process:** Any process that changes the data, producing an output. It might perform computations, sort data based on logic, or direct the data flow based on business rules. A short label describes the process, such as "Submit payment."



- **Data Store:** Files or repositories with information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders”.



- **Data Flow:** The route data takes between the external entities, processes, and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”



### 5.3.1 Data Flow diagram for patient register:

#### 5.3.2 Level 0:

A context diagram is a top-level (also known as "Level 0") data flow diagram. It only contains one process node ("Process 0") that generalizes the function of the entire system in relation to external entities. Draw data flow diagrams can be made in several nested layers.

#### 5.3.3 Level 1:

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the system's main functions, as you break down the high-level process of the Context Diagram into its subprocesses.

### 5.3.4 Level 2:

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.

## 5.4 ER Diagram:

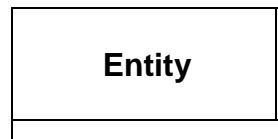
ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help explain databases' logical structure. ER diagrams are created based on three basic concepts: entities, attributes, and relations

An ER diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

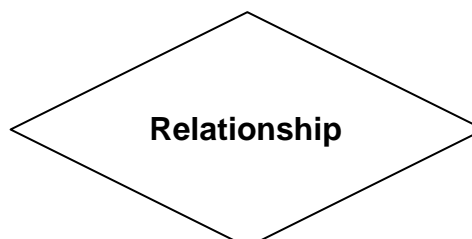
- **Entities**, which are represented by rectangles. An entity is an object or concept you want to store information about.



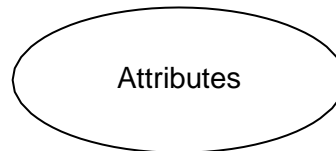
- weak entity is an entity that must be defined by a foreign key relationship with another entity, as it cannot be uniquely identified by its own attributes alone.



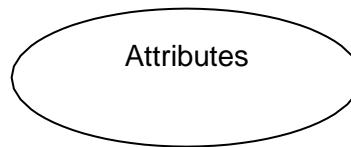
- **Actions**, represented by diamond shapes, show how two entities share information in the database. In some cases, entities can be self-linked. For example, employees can supervise other employees.



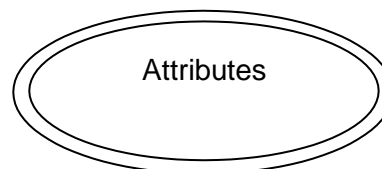
- **Attributes** are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

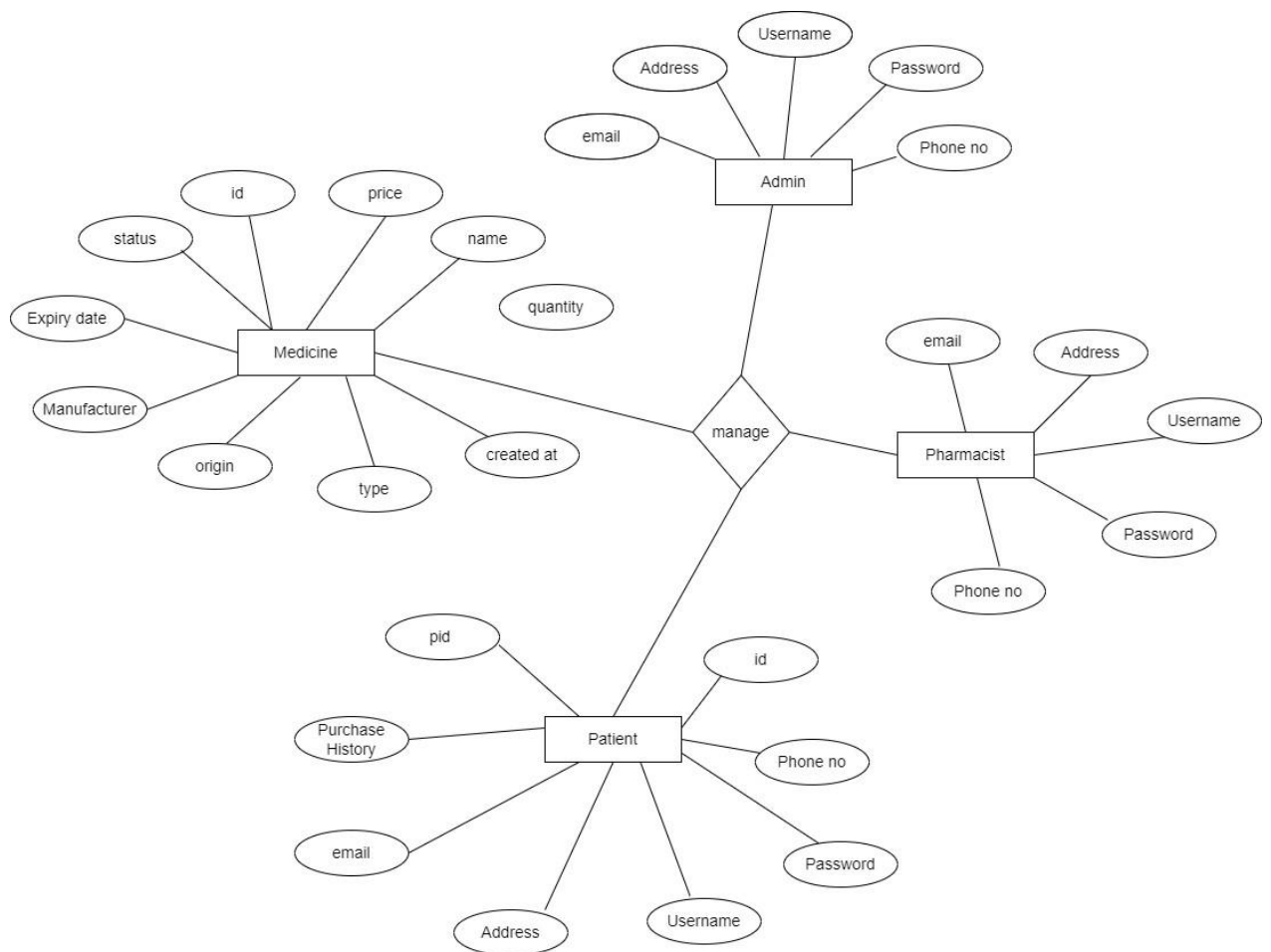


- A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.



- A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.





5.4 Figure: E-R diagra

### 5.5 ACTIVITY DIAGRAM:

The activity diagram used to describe flow of activity through a series of actions. Activity diagram is a important diagram to describe the system. An activity diagram shows the overall flow of control.

#### ➤ Initial State or Start Point

A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram. For activity diagram using swim lanes, make sure the start point is placed in the top left corner of the first column.



➤ **Activity or Action State**

An action state represents the non-interruptible action of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.



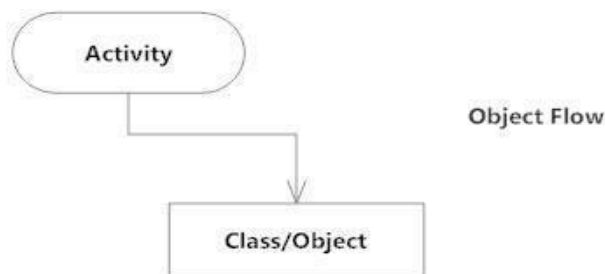
➤ **Action Flow**

Action flows, also called edges and paths, illustrate the transitions from one action state to another. They are usually drawn with an arrowed line.



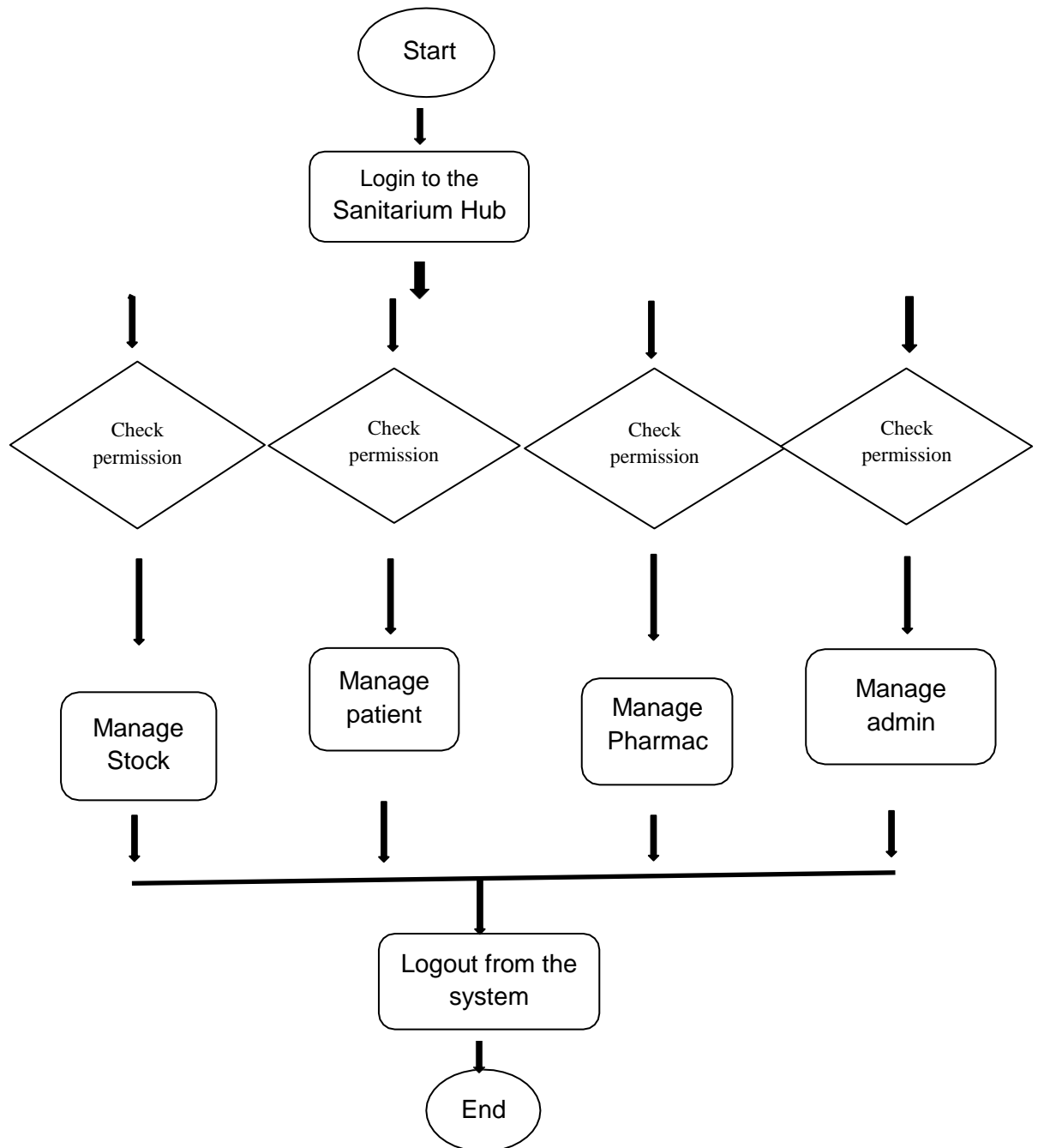
➤ **Object Flow**

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



➤ **Decisions and Branching**

A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



5.5 Figure: Activity diagram of sanitarium hub

**CHAPTER 6**  
**REQUIREMENT SPECIFICATION**



## **6.1 INTRODUCTION:**

To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

## **6.2 HARDWARE REQUIREMENTS:**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

### **6.2.1 HARDWARE REQUIREMENTS FOR PRESENT PROJECT:**

PROCESSOR: Intel Core i7

RAM: 8 GB

HARD DISK: 250 GB

## **6.3 SOFTWARE REQUIREMENTS:**

Software Requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

### **6.3.1 SOFTWARE REQUIREMENTS FOR PRESENT PROJECT:**

Operating System : Windows 10  
Front End : HTML, CSS, JavaScript.  
Server Side Script : PHP, Laravel  
Database : MySQL

## **6.4 SOFTWARE SPECIFICATION**

### **6.4.1 HTML:**

HTML or Hypertext Markup Language is the standard markup language used to create web pages.

HTML is written in the form of HTML elements consisting of *tags* enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent *empty elements* and so are unpaired, for example <img>. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*). Though not always necessary, it is best practice to append a slash to tags which are not paired with a closing tag.

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret

The content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

#### **6.4.2 CASCADING STYLE SHEETS (CSS):**

It is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content.

CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed.

While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified. However if the author or the reader did not link the document to a specific style sheet the default style of the browser will be applied.

#### **6.4.3 MySQL:**

MySQL is developed, distributed, and supported by Oracle Corporation. MySQL is a database system used on the web that runs on a server. MySQL is ideal for both small and large applications. It is very fast, reliable, and easy to use. It supports standard SQL. MySQL can be compiled on a number of platforms.

The data in MySQL is stored in tables. A table is a collection of related data, and it consists of columns and rows. Databases are useful when storing information categorically.

#### **6.4.4 FEATURES OF MySQL:**

- Internals and portability:
- Written in C and C++.
- Tested with a broad range of different compilers.
- Works on many different platforms.
- Tested with Purify (a commercial memory leakage detector) as well as with Val grind, a GPL tool.
- Uses multi-layered server design with independent modules.

#### **6.4.5 Security:**

- A privilege and password system that is very flexible and secure, and that enables host-based verification.
- Password security by encryption of all password traffic when you connect to a server.

#### **6.4.6 Scalability and Limits:**

- Support for large databases. We use MySQL Server with databases that contain 50 million records. We also know of users who use MySQL Server with 200,000 tables and about 5,000,000,000 rows.
- Support for up to 64 indexes per table. Each index may consist of 1 to 16 columns or parts of columns. The maximum index width is 767 bytes for InnoDB tables, or 1000 for MySQL; before MySQL 4.1.2, the limit was 500 bytes. An index may use a prefix of a column for CHAR, VARCHAR, BLOB, or TEXT column types.

#### **6.4.7 Connectivity:**

- Clients can connect using TCP/IP sockets on any platform.
- On UNIX systems, clients can connect using UNIX domain socket files.

#### **6.4.8 Localization:**

- The server can provide error messages to clients in many languages.
- All data is saved in the chosen character set.

#### **6.4.9 Clients and Tools:**

- MySQL includes several client and utility programs. These include both command-line programs such as MySQL dump and MySQL admin, and graphical programs such as MySQL Workbench.
- MySQL Server has built-in support for SQL statements to check, optimize, and repair tables. These statements are available from the command line through the MySQL check client. MySQL also includes MySQL check, a very fast command-line utility for performing these operations on MySQL tables.
- MySQL programs can be invoked with the --help or -? Option to obtain online assistance.

#### **6.4.10 WHY TO USE MySQL:**

- Leading open-source RDBMS
- Ease of use – No frills
- Fast
- Robust
- Security
- Multiple OS support
- Free
- Technical support
- Support large database– up to 50 million rows, file size limits up to 8 million TB

#### **6.4.11 JAVASCRIPT:**

JavaScript is the scripting language of the Web. All modern HTML pages are using JavaScript. A scripting language is a lightweight programming language. JavaScript code can be inserted into any HTML page, and it can be executed by all types of web browsers. JavaScript is easy to learn.

#### **6.4.12 WHY TO USE JAVASCRIPT:**

- HTML to define the content of web pages
- CSS to specify the layout of web pages
- JavaScript to specify the behavior of web pages

#### **6.4.13 OTHER USES OF JAVASCRIPT:**

- Delete HTML elements
- Create new HTML elements
- Copy HTML elements
- In HTML, JavaScript is a sequence of statements that can be executed by the web browser.

#### **6.4.14 PHP:**

- PHP is an acronym for "PHP Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP costs nothing, it is free to download and use

#### **6.4.15 WHY PHP?**

- PHP runs on various platforms (Windows, Linux, UNIX, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: [www.php.net](http://www.php.net)

**CHAPTER 7**  
**CODE IMPLEMENTATION**

## 7.1 INTRODUCTION

The implementation of the Medicine Management System – a cutting-edge solution designed to streamline and enhance the process of managing medications efficiently and accurately. In an era where the healthcare industry is rapidly evolving, the significance of seamless medication management cannot be overstated. This software system serves as a comprehensive platform that empowers healthcare facilities, pharmacies, and patients alike to exercise precise control over medication-related tasks.

Throughout this code implementation, we will explore the various facets of the Medicine Management System. From a user-friendly interface for healthcare professionals to manage prescriptions, dosages, and patient records, to an intuitive platform for patients to track their medication schedules, this system aims to bridge the gap between medical professionals and those under their care.

## 7.2 Website Frontend

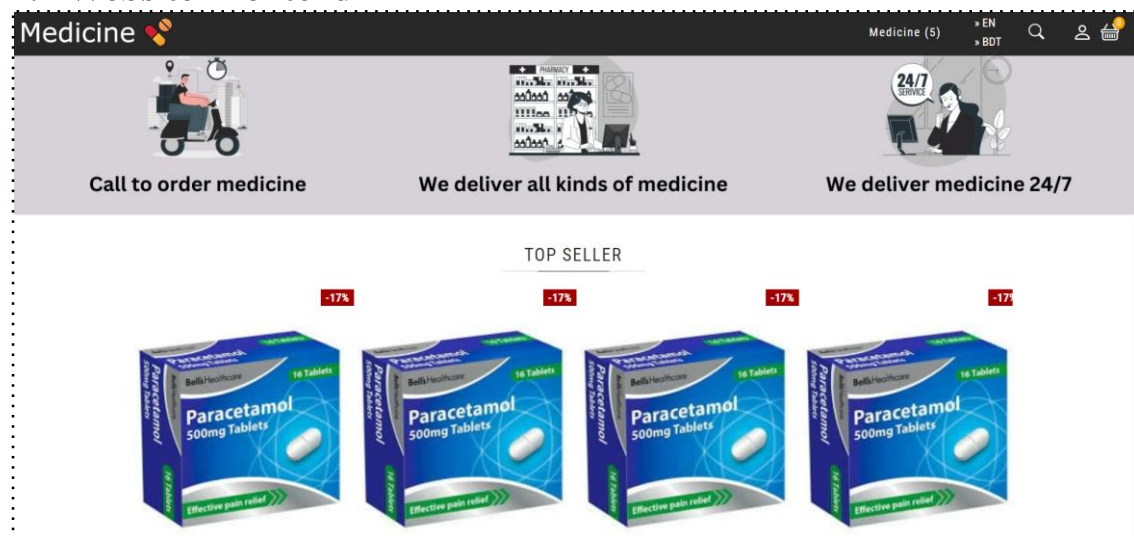


Figure 7.1: Website frontend home page



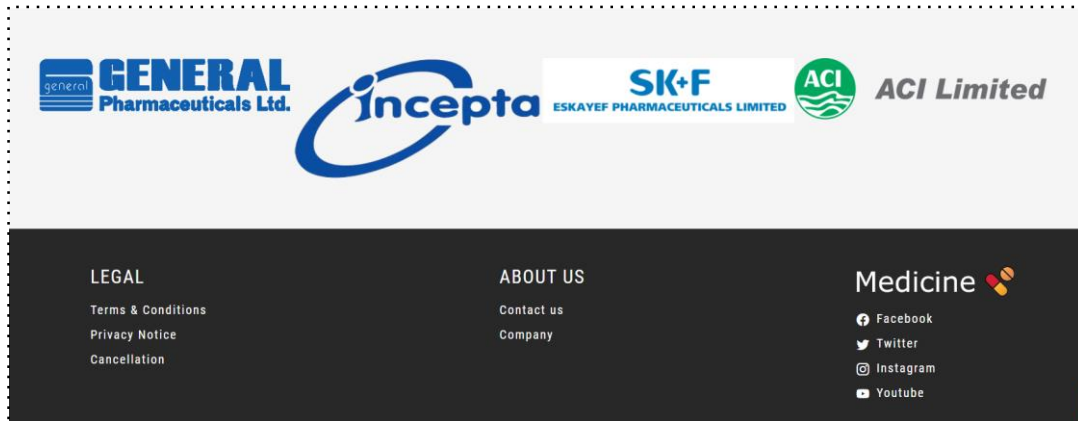


Figure 7.2: Website frontend footer



Figure 7.3: Medicine product section

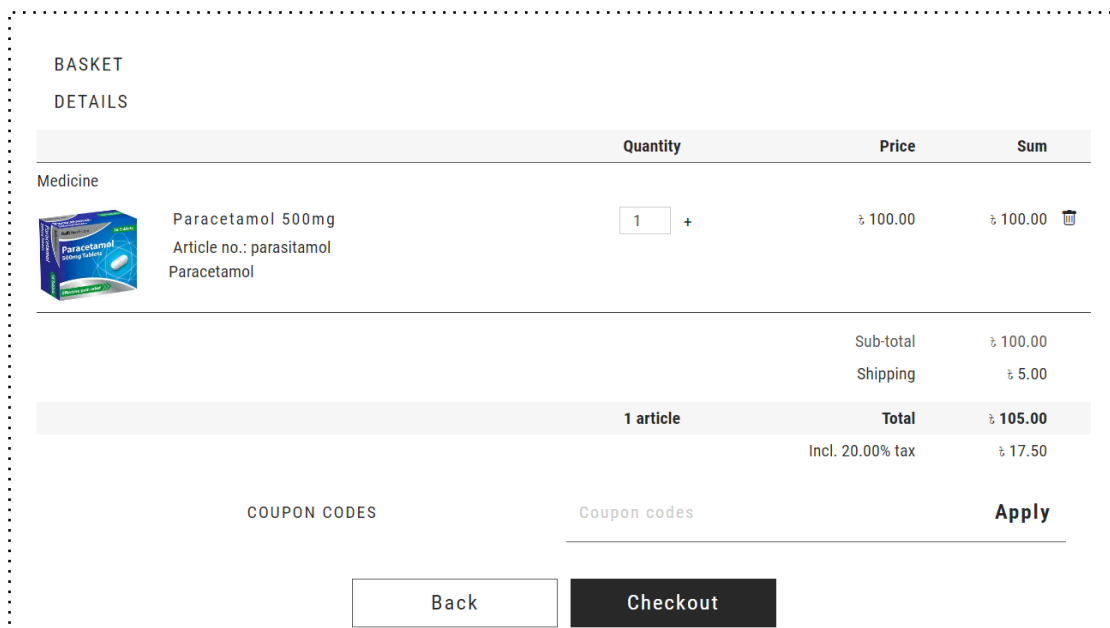


Figure 7.4: Medicine product cart section

Figure 7.5: Medicine product shipping address section

Figure 7.5: Medicine product shipping delivery section

Figure 7.5: Medicine product payment section with multiple options

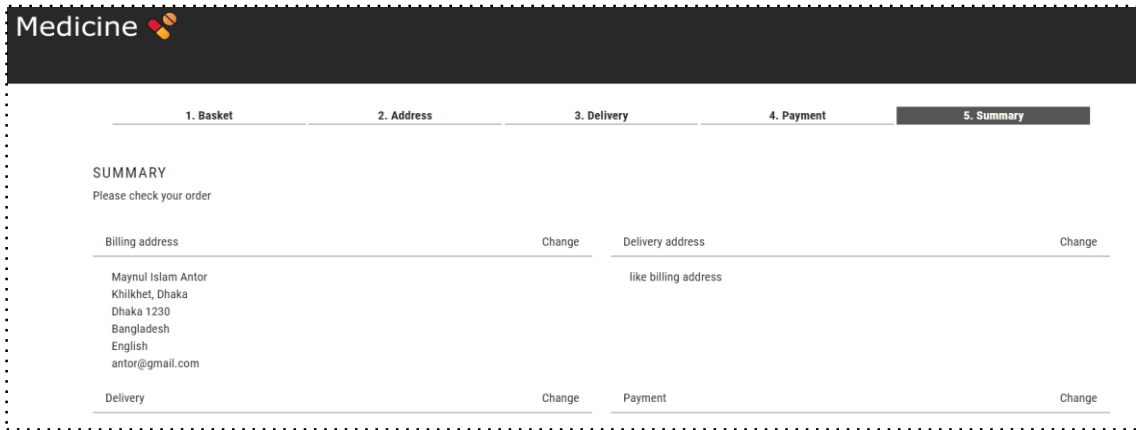


Figure 7.6: Medicine product summary section

### 7.3 Website Backend

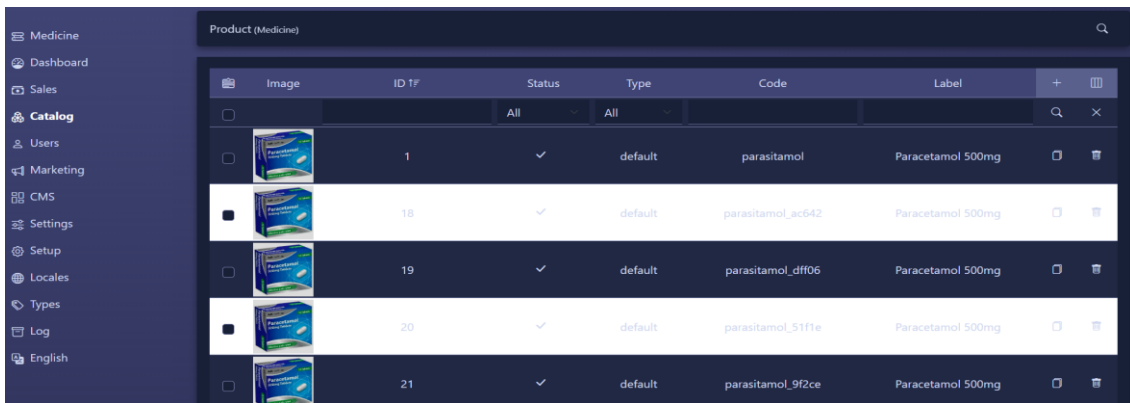


Figure 7.7: Website backend management console.

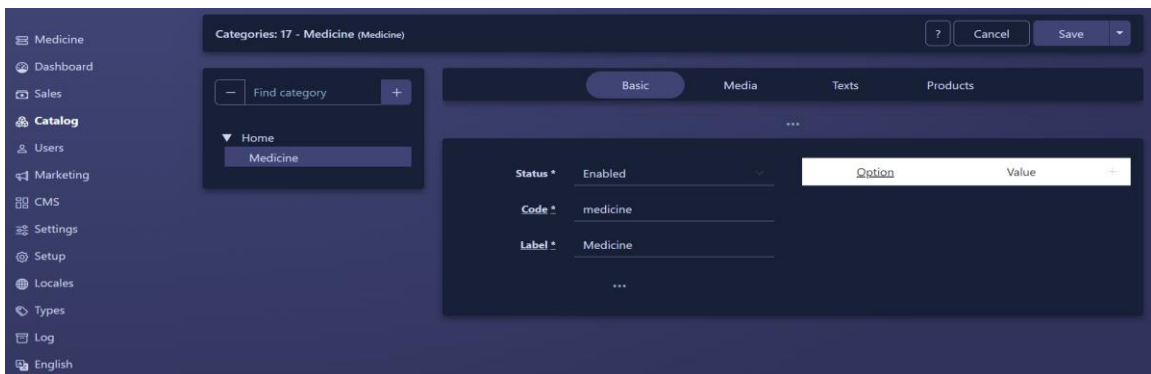
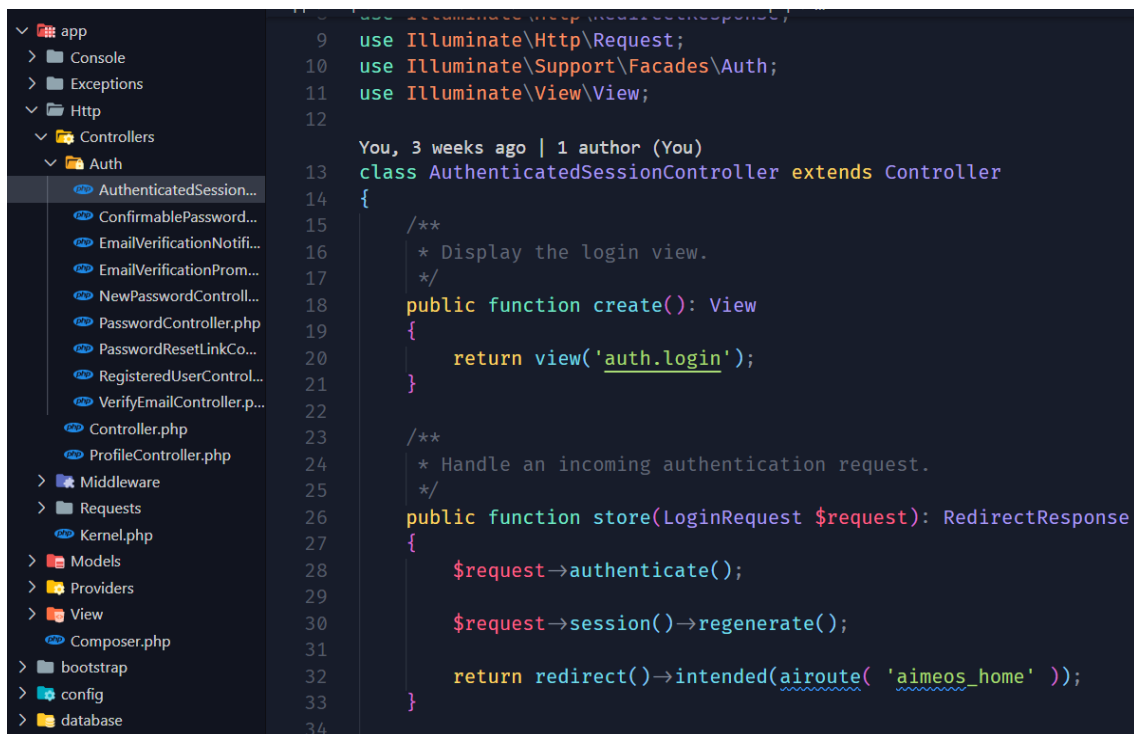


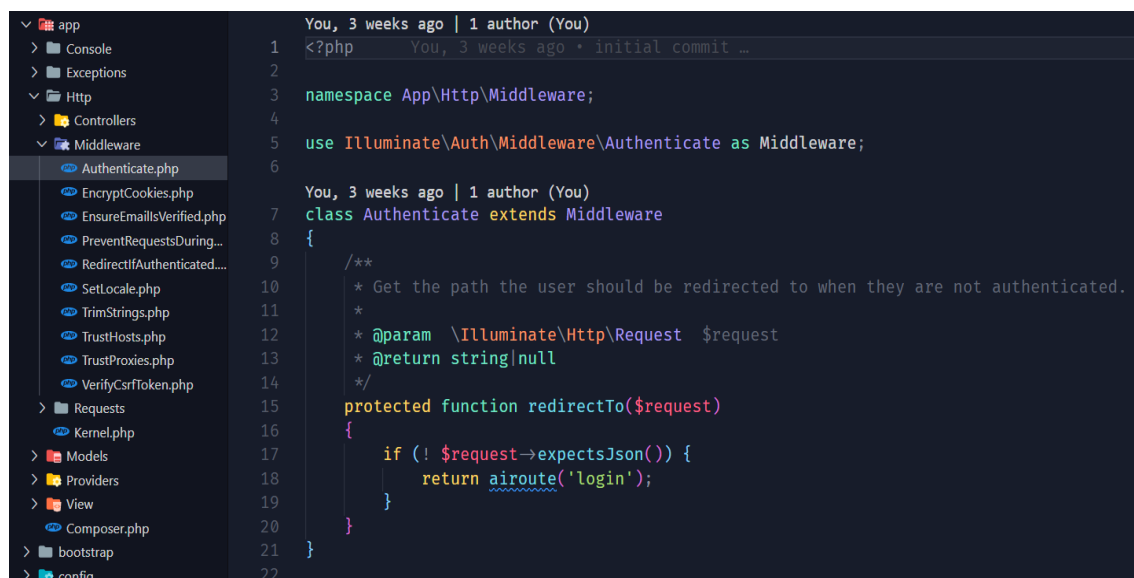
Figure 7.7: Website backend product category CRUD section.

## 7.4 Code Implementation



```
9 use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Auth;
11 use Illuminate\View\View;
12
13 You, 3 weeks ago | 1 author (You)
14 class AuthenticatedSessionController extends Controller
15 {
16     /**
17      * Display the login view.
18      */
19     public function create(): View
20     {
21         return view('auth.login');
22     }
23
24     /**
25      * Handle an incoming authentication request.
26      */
27     public function store(LoginRequest $request): RedirectResponse
28     {
29         $request->authenticate();
30
31         $request->session()->regenerate();
32
33         return redirect()->intended(airoute( 'aimeos_home' ));
34     }
35 }
```

Figure 7.8: Project Auth Controller



```
1 You, 3 weeks ago | 1 author (You)
2 <?php You, 3 weeks ago * initial commit ...
3 namespace App\Http\Middleware;
4
5 use Illuminate\Auth\Middleware\Authenticate as Middleware;
6
7 You, 3 weeks ago | 1 author (You)
8 class Authenticate extends Middleware
9 {
10     /**
11      * Get the path the user should be redirected to when they are not authenticated.
12      *
13      * @param \Illuminate\Http\Request $request
14      * @return string|null
15      */
16     protected function redirectTo($request)
17     {
18         if (!$request->expectsJson()) {
19             return airoute('login');
20         }
21     }
22 }
```

Figure 7.9: Project Auth Middleware

```

9 use Laravel\Sanctum\HasApiTokens;
10
11 You, 3 weeks ago | 1 author (You)
12 class User extends Authenticatable implements MustVerifyEmail
13 {
14     use HasApiTokens, HasFactory, Notifiable;
15
16     /**
17      * The attributes that are mass assignable.
18      *
19      * @var array<int, string>
20      */
21     protected $fillable = [
22         'name',
23         'email',
24         'password',
25         'siteid',
26     ];
27
28     /**
29      * The attributes that should be hidden for serialization.
30      *
31      * @var array<int, string>
32      */
33     protected $hidden = [
34         'password',
35         'remember_token',
36     ];

```

Figure 7.10: User Database Model

```

1 <!DOCTYPE html>
2 <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
3
4 <head>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <meta name="csrf-token" content="{{ csrf_token() }}">
8
9     <title>{{ config('app.name', 'Laravel') }}</title>
10
11     <!-- Fonts -->
12     <link rel="preconnect" href="https://fonts.bunny.net">
13     <link href="https://fonts.bunny.net/css?family=figtree:400,500,600&display=swap" rel="stylesheet" />
14
15     <!-- Scripts -->
16     @vite(['resources/css/app.css', 'resources/js/app.js'])
17 </head>
18 <body class="font-sans antialiased">
19     <div class="min-h-screen bg-gray-100 dark:bg-gray-900">
20         @include('layouts.navigation')
21
22         <!-- Page Heading -->
23         @if (isset($header))
24             <header class="bg-white dark:bg-gray-800 shadow">
25                 <div class="max-w-7xl mx-auto py-6 px-4 sm:px-6 lg:px-8">
26                     {{ $header }}
27                 </div>
28             </header>
29         @endif

```

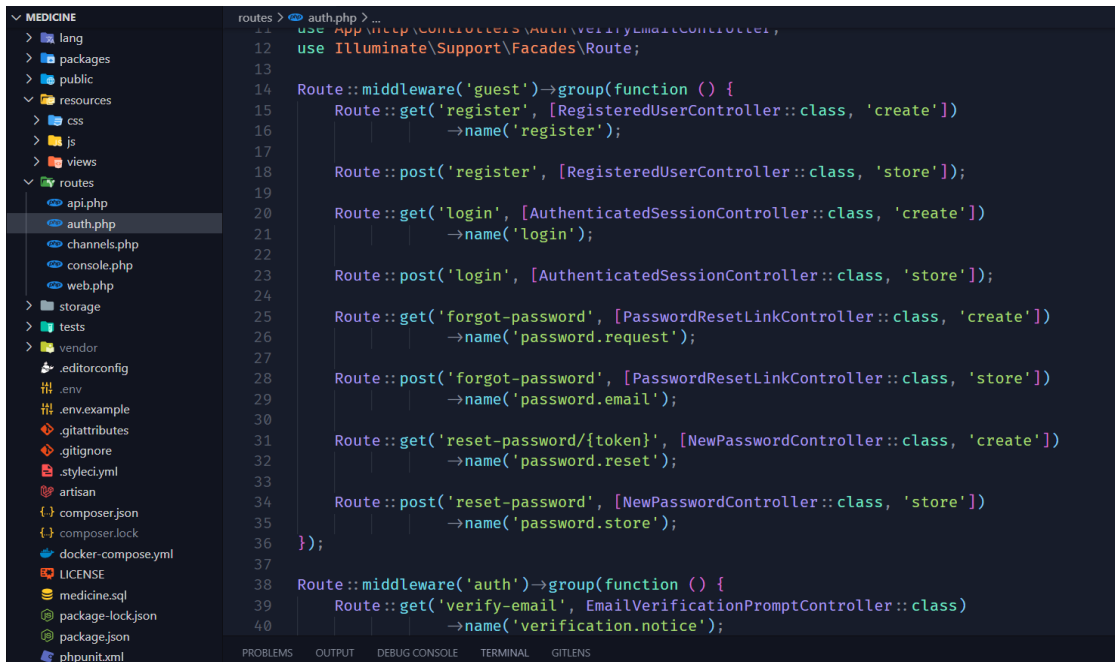
Figure 7.11: App view layout

```

1 <x-app-layout>
2     <x-slot name="header">
3         <h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">
4             {{ __('Dashboard') }}
5         </h2>
6     </x-slot>
7
8     <div class="py-12">
9         <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
10            <div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">
11                <div class="p-6 text-gray-900 dark:text-gray-100">
12                    {{ __('You're logged in!') }}
13                </div>
14            </div>
15        </div>
16    </div>
17 </x-app-layout>

```

Figure 7.12: App dashboard view layout



```
1 use App\Http\Controllers\Auth\VerifyEmailController;
2 use Illuminate\Support\Facades\Route;
3
4
5
6
7
8
9
10
11
12
13
14 Route::middleware('guest')->group(function () {
15     Route::get('register', [RegisteredUserController::class, 'create'])
16         ->name('register');
17
18     Route::post('register', [RegisteredUserController::class, 'store']);
19
20     Route::get('login', [AuthenticatedSessionController::class, 'create'])
21         ->name('login');
22
23     Route::post('login', [AuthenticatedSessionController::class, 'store']);
24
25     Route::get('forgot-password', [PasswordResetLinkController::class, 'create'])
26         ->name('password.request');
27
28     Route::post('forgot-password', [PasswordResetLinkController::class, 'store'])
29         ->name('password.email');
30
31     Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])
32         ->name('password.reset');
33
34     Route::post('reset-password', [NewPasswordController::class, 'store'])
35         ->name('password.store');
36 });
37
38 Route::middleware('auth')->group(function () {
39     Route::get('verify-email', EmailVerificationPromptController::class)
40         ->name('verification.notice');
```

Figure 7.13: Project all routes files

# CHAPTER 8

# CONCLUSION

### **8.1 Conclusion:**

This project has been a rewarding experience in more than one way. The entire project work has enlightened us in the following areas. We have gained an insight into the working of the medicine. This represents a typical real-world situation. Our understanding of database design has been strengthened this is because in order to generate the final reports of database designing has to be properly followed. Scheduling a project and adhering to that schedule creates a strong sense of time management. Sense of teamwork has developed and confidence of handling real life project has increased to a great extent.

Initially, there were problem with the validation but with discussions, we were to implement validations.

### **8.2 Limitations of the system**

- Online payment is not available at this version.
- Data delete & edit system is not available for all section.
- User account not verified by Mobile SMS not available in this system.
- Loss of data due to mismanagement.

### **8.3 Future Enhancement:**

The proposed system is Web-based online medicine service system. We can enhance this system by including more facilities like delivery system for the weak and sick patient. Providing such features enable the users to include more comments into the system.



## **Reference:**

- Agile Software Development Ecosystems, By James A. Highsmith, Jim Highsmith, publishing date 2002
- Pharmacy Management Software for Pharmacy Technicians: a Worktext, By Daa Enterprises, Inc, publishing date 2017
- Systems Analysis and Design, By Alan Dennis, Barbara Wixom, Roberta M. Roth, publishing date 2021
- Software Requirements & Specifications: A Lexicon of Practice, Principles, and Prejudices By Michael Jackson, Michael James Jackson, publishing date 1995