A Project Report

On

CARE&CURE HOSPITAL

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

BY

SYEDA AIMEN FATIMA **160319733011**

Under the guidance

Of

Ms. Afroze Begum

Assistant Professor

Department Of CSE, DCET

Department of Computer Science Engineering

Deccan College of Engineering and Technology

(Affiliated to Osmania University) Hyderabad

CERTIFICATE

This is to certify that the project report entitled Care&Cure hospital submitted by Ms. Ashna Manzoor (160319733002), Ms. Syeda Aimen Fatima (160319733011) and Ms. Amina Amatur Rahman Siddiqui (160319733014) in partial fulfillment for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering by the Osmania University is a record of Bonafide work carried out by them under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Internal Guide	Head of the Department
Mrs. Afroze Begum Raziuddin	Dr. Syed
Assistant Professor	Professor & Head
Department of CSE	Department of CSE
DCET, Hyderabad	DCET, Hyderabad

Declaration

This is to certify that the work reported in the present project entitled **"CARE&CURE HOSPITALS"** is a record of work done by me in the Department of Computer Science & Engineering, Deccan College of Engineering and Technology, Osmania University, Hyderabad in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Computer Science & Engineering.

The results presented in this dissertation have been verified and are found to be satisfactory. The results embodied in this dissertation have not been submitted to any other university for the award of any degree or diploma.

Acknowledgement

I am thankful to Principal **Dr M.A. MALIK** for providing excellent infrastructure and a nice atmosphere for completing this project successfully.

I am thankful to Head of the department Dr. **SYED RAZIUDDIN** for providing the necessary facilities during the execution of our project work.

I would like to express my sincere gratitude and indebtedness to my project supervisor **Mrs. Afroze Begum for** her valuable suggestions and interest throughout the course of this project.

This project would not have been a success without my internal guide. So, I would extend my deep sense of gratitude to my internal guide **Mrs. Afroze Begum,** for the effort she took in guiding me in all the stages of completion of my project work. We also thank for her valuable suggestions, advice, guidance and constructive ideas in each and every step, which was indeed a great need towards the successful completion of the project.

I convey my heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

Abstract

Hospital management is the process where health care providers effectively and efficiently administer everything from patient to managing the hospital details.

Despite the advancement and evolution of technology, there still exist certain problems faced by management authorities in implementing it among them, the technical and human challenges are considered to be complicated factors.

In our project we have tried to overcome and simplify few features and made it user friendly so that its implementation in hospitals is easier for healthcare faculty like appointment scheduling, medicine billing, inpatient & outpatient details & keeping record of all the performed surgeries.

List of Tables

List of figures

Fig. 7.14 Add patient page (for patients' access)

Fig. 7.15 View Doctor page (for patients' access)

Fig 7.16 Contact page (for patients' access)

List of Abbreviations

1. GUI – Graphical User Interface
2. SQL-Structured Query Language
3. GUI- Graphical User Interface
4. AWT- Abstract Window Toolkit
5. IDE- Integrated Development Environment
6. DFD- Data Flow Diagrams
7. WBS-Work Breakdown Structure

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Introduction

Hospital is the essential part of our lives, providing best medical facilities to people suffering from various ailments, which may be due to change in climate conditions, increased workload, emotional trauma stress etc. It is necessary for the hospitals to keep track of its day-today activities & records of its patients, doctors and other staff personals that keep the hospital running smoothly & successfully.

But keeping track of all the activities and their records on paper is very cumbersome and error prone. It also is very inefficient and a time-consuming process observing the continuous increase in the population and a number of people visiting the hospital. Recording and maintaining all these records are highly unreliable, inefficient and error prone. It is also not economically & technically feasible to maintain these records on paper.

Thus, keeping the working of the manual system as the basis of the project, we have developed an automated version of the manual system, named as "Care&Cure hospital". The main aim of our project is to provide a paper-less hospital up to 90%. It also aims at providing low-cost reliable automation of the existing systems. The system also provides excellent security of data at every level of user-system interaction and also provides robust and reliable storage and backup facilities.

1.2 Problem Statement

Care&Cure hospital is being used for decades. Most of the hospitals in India make use of HMS but they face certain challenges in implementing it. Among them, technical and human challenges are considered to be the complicated factors while implementing HMS.

Human Challenges: Awareness of HMS advantages & importance. In general, Experience, and knowledge of using computer applications. Impressions and beliefs regarding HMS and making use of them efficiently.

Technical Challenges: Other few technical challenges that fail the implementation of HMS in the healthcare industry that includes Networks and Computer have different maintenance problems, lack of no standards for Data entry and Data retrieval, difficulties in training users technically to use HMS.

1.3 Objective

The project Care&Cure hospital‖ is aimed to develop to maintain the day-to-day state of admission/discharge of patients, list of doctors & patients, list of appointments, etc.

It is designed to achieve the following objectives.

(a) To computerize all details regarding patient & doctors as well as the hospital.

(b) Scheduling the appointment of patient with doctors to make it convenient for both.

(c) The information must be kept up to date and their record should be kept in the system for historical purpose.

(d) Reduce the paper work and provide the 24/7 accessible website with user-friendly GUI (Graphical User Interface) to manage the records.

1.4 Scope

The scope of the project can be summarized as follows: The proposed software product is the "Care&Cure hospital". The system will be used in any Hospital to get the information from the patients and then storing that data for future usage.

The current system in use is a paper-based system. It is too slow and cannot provide updated lists of patients within a reasonable timeframe. The intention of the system is to reduce over-time pay and increase the number of patients that can be treated accurately**.**

1.5 Infrastructure

Based on information communication technology, as well as wireless and mobile communication protocols, an intelligent healthcare infrastructure builds efficiency across an organization.

Acting as the central nervous system of a hospital, the infrastructure integrates and enables communication between traditionally disparate systems, such as power, building management, security and IT.

Through wireless communication, an intelligent technology infrastructure enhances overall patient experience and hospital efficiency. It is a Web based application developed in windows environment using PYTHON and also supports MySQL database

CHAPTER 2

LITERATURE SURVEY

2.1 Previous Studies

HMS was introduced to solve the complications coming from managing all the paper works of every patient associated with the various departments of hospitalization with confidentiality. HMS provides the ability to manage all the paperwork in one place, reducing the work of staff in arranging and analyzing the paperwork of the patients. HMS does many works like:

- Maintain the medical records of the patient
- Maintain the contact details of the patient
- Keep track of the appointment dates
- Save the insurance information for later reference
- Tracking the bill payments.

The advantages of HMS can be pinpointed to the following:

- Time-saving Technology
- Improved Efficiency by avoiding human errors
- Reduces scope for Error
- Data security and correct data retrieval made possible
- Cost effective and easily manageable
- Easy access to patient data with correct patient history
- Improved patient care made possible
- Easy monitoring of supplies in inventory
- Reduces the work of documentation
- Better Audit controls and policy compliance.

2.2 Technology Studied

HTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

HTML is a markup language used by the browser to manipulate text, images, and other content, in order to display it in the required format. HTML was created by Tim Berners-Lee in 1991. The first-ever version of HTML was HTML 1.0, but the first standard version was HTML 2.0, published in 1999.

Elements and Tags: HTML uses predefined tags and elements which tell the browser how to properly display the content. Remember to include closing tags. If omitted, the browser applies the effect of the opening tag until the end of the page.

Features of HTML:

- It is easy to learn and easy to use.
- It is platform-independent.
- Images, videos, and audio can be added to a web page.
- Hypertext can be added to the text.
- It is a markup language.

Why should you learn HTML?

- It is a simple markup language. Its implementation is easy.
- It is used to create a website.
- Helps in developing fundamentals about web programming.

- Boost professional career.

Advantages:

- HTML is used to build websites.
- It is supported by all browsers.
- It can be integrated with other languages like CSS, JavaScript, etc.

Disadvantages:

- HTML can only create static web pages. For dynamic web pages, other languages have to be used.
- A large amount of code has to be written to create a simple web page.
- The security feature is not good

PYTHON

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. We don't need to declare the type of variable because it is a dynamically typed language.

For example, x = 10. Here, x can be anything such as String, int, etc.

Features

1. Easy to code.

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

2. Free and Open Source.

Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword.

3. Object-Oriented Language

One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

4. GUI Programming Support

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

5. High-Level Language

Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6. Extensible feature

Python is an Extensible language. We can write us some Python code into C or C++ language and also, we can compile that code in C/C++ language.

7. Python is Portable language

Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

8. Python is Integrated language

Python is also an integrated language because we can easily integrated python with other languages like C, C++, etc.

9. Interpreted Language

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode.

10. Large Standard Library

Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.

MYSQL DATA BASE

MySQL is a relational database management system (RDBMS) based on the SQL (Structured Query Language) queries. It is one of the most popular languages for accessing and managing the records in the table. MySQL is open-source and free software under the GNU license. Oracle Company supports it.

Features

1. Relational Database Management System (RDBMS). MySQL is a relational database management system. This database language is based on the SQL queries to access and manage the records of the table.
2. Easy to use. MySQL is easy to use. We have to get only the basic knowledge of SQL. We can build and interact with MySQL by using only a few simple SQL statements.
3. It is secure. MySQL consists of a solid data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.
4. Client/ Server Architecture. MySQL follows the working of a client/server architecture. There is a database server (MySQL) and arbitrarily many clients

(application programs), which communicate with the server; that is, they can query data, save changes, etc.

5. Free to download. MySQL is free to use so that we can download it from MySQL official website without any cost.
6. It is scalable. MySQL supports multi-threading that makes it easily scalable. It can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, we can increase this number to a theoretical limit of 8 TB of data.
7. Speed. MySQL is considered one of the very fast database languages, backed by a large number of the benchmark test.
8. High Flexibility. MySQL supports a large number of embedded applications, which makes MySQL very flexible.
9. Compatible on many operating systems. MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

CHAPTER 3

SYSTEM ANALYSIS

3.1 SDLC

Software Development Life Cycle (SDLC). There are various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as ―Software Development Process Models‖ (e.g. Waterfall model, Incremental model, V-model, iterative model, etc.). Each process model follows a particular life cycle in order to ensure success in process of software development.

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase.

After coding and development, the testing verifies the deliverable of the implementation phase against requirements. 3.1.1 Life Cycle Phases.

There are following six phases in every Software development life cycle model: Requirement gathering and analysis, Design, Implementation or coding, Testing, Deployment, Maintenance.

Requirement gathering and analysis

Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements like; who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirement gathering phase.

After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also

studied. Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

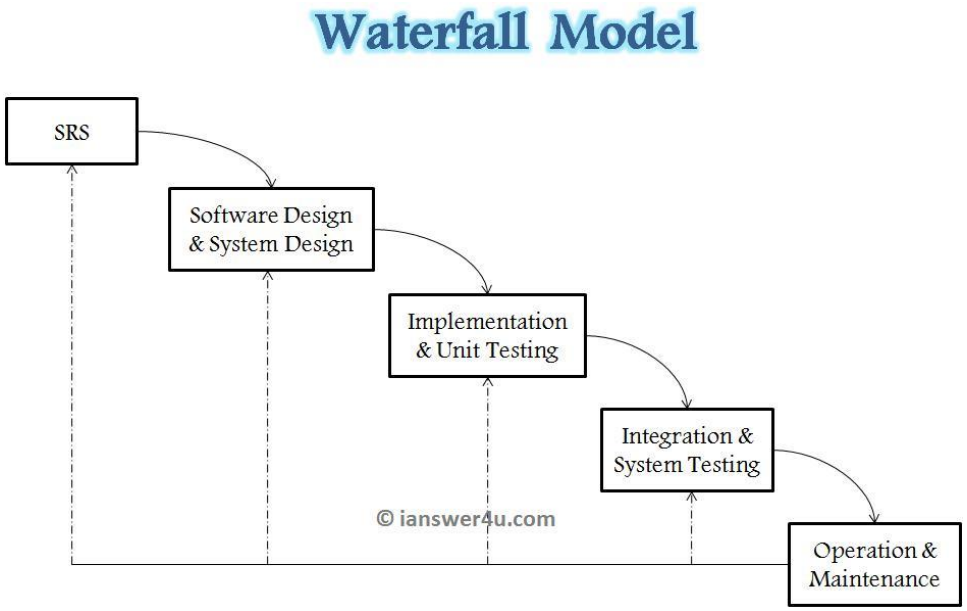


Fig. 3.1 Waterfall Model

Design

In this phase the system and software design are prepared from the requirement specifications which were studied in the first phase.

System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

Implementation / Coding:

On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

Testing: After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase unit testing, integration testing, system testing, acceptance testing is done.

Deployment: After successful testing the product is delivered / deployed to the customer for their use.

Maintenance: Once when the customers start using the developed system then the actual problems come up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

3.2 Existing System

Before computerized Care&Cure hospital came into practice, it was difficult to keep proper records of the daily activities of hospitals, patient information, maintenance schedule of equipment's in the hospital, and how funds are being allocated and used. This resulted in waste of money, time and manpower. Care&Cure hospital is an information management system designed to help manage the various aspects of a hospital (administrative, clinical and financial). It helps in monitoring and controlling the hospital's daily transactions, as well as the hospital's performance. It also helps to address the critical requirements of the hospital. Care&Cure hospital enables access to the right information and automation of complex task, thereby allowing staff to spend more time caring for patients. Care&Cure hospital is custom built to meet the specific requirements of the medium and large size hospitals across the globe1. Current Management System Most hospitals face several challenges with Care&Cure hospital because some of them are still using manual processes, while the ones that use the computerized method are also faced with the challenge of adjusting to it. Such problems include

- High cost of software development, deployment and improvement.

· Difficulty in migrating from manual processes, because both staff and patients are used to the manual

· Processes and so are unable to speedily cope with the new system.

· Lack of IT friendly medical personnel is also presenting several challenges.

· Huge influx of patients visiting government hospitals makes the process of migrating to automated

· Processes highly difficult. They do not have the patience to wait for registration and data entry and often fail to understand the functioning of automated processes.

Limitations of existing system

Lack of security of data. Time consuming. Consumes large volume of paper work. Manual work. No direct role for the higher officials. To avoid all these limitations and make the system working more accurately it needs to be computerized.

3.3 Purpose System

Before starting to develop any software, it is important to define what the problem is and what the client requirement is. Though there has been a lot of advancement made in the past decade in a Care&Cure hospital, there still exists problems that are required to be rectified.

The problems in the existing system are as follows:

1. Waiting time: Patients that have an appointment on a given still time has to wait for hours to meet the doctor.
2. Doctor’s availability and timings: There are a number of doctors available at the hospital. Still patients face problems to meet a doctor despite having an appointment scheduled.
3. In-Patient Out-patient details: Details recording is required for all the patients that have come to the hospital for visit and admission for future use.
4. Record of previous data: All the previous information and associations carried out in the hospital are required to be recorded including previous surgeries performed.
5. Medical billing: At the pharmacy, the payment requires the checking price and quantity of item purchased and totaling the amount which takes time and there are chances of human errors in reading price and totaling amount.
6. Cost effective: The poor and old, can’t afford the huge amount of money spent on hospital bills.

The desired system state is as follows:

1. Appointment scheduling: If there was some appointment time scheduled the patients shall be able to meet the doctor at the promised time.
2. Doctors' availability and timings: All the working doctors in a hospital must be scheduled patients in an effective considering doctors’ workload and patients waiting time.

3. In-patient Out-patient details: All patients details must be recorded considering the health issues that may arise in the future which may help in the future diagnosis.
4. Record of previous data: All the information about the hospital including the patients and staff that are/were previously associated with the hospital since its establishment must be recorded as a proof of hospital's existence and for any future use to solve upcoming problems in the future.
5. Medical billing: A computerized medical billing format should be used which decreases the time and human effort required in the selling and purchase of items.
6. Health card: For widows and old people, health cards must be issued as an act of kindness.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 Feasibility Study

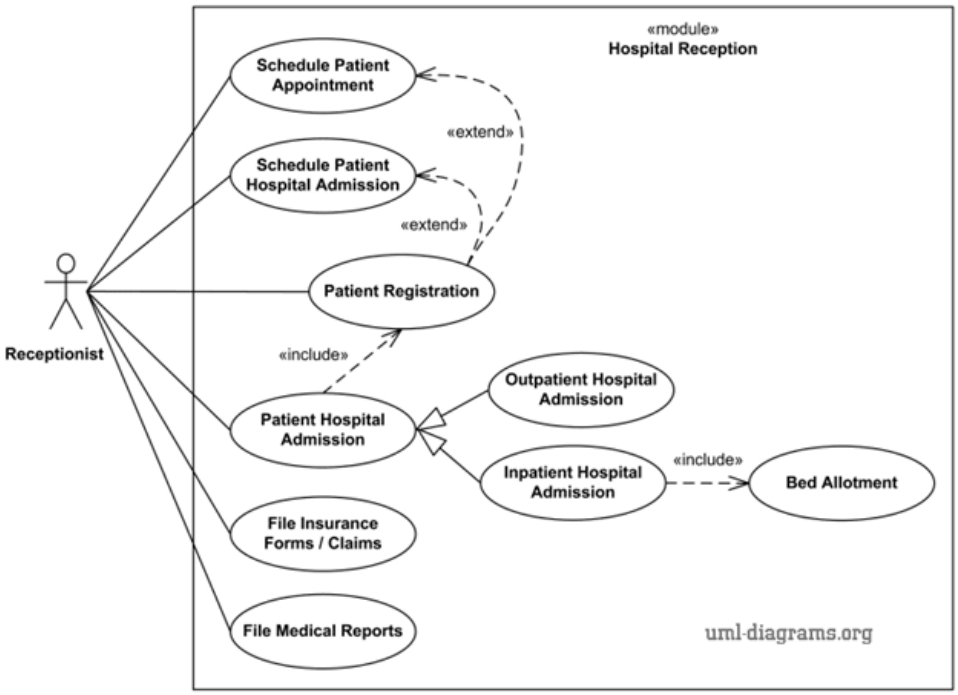
The Benefits of Conducting a Hospital Feasibility Study

- Broadening the scope in order to identify the new opportunities
- Enhances the resources skills while anticipating the challenges
- Helps in narrowing down the potential risks and in finding solutions
- Most importantly, identifies the outcome of the project with utmost accuracy

Here are the areas that a healthcare consultant could help you in deriving the accurate data with effective feasible study.

- **Financial Feasibility –** The assessment typically analyzes the economical scope of the project including the investments and the return on investments and the time it would take to reach the set target. The healthcare analysts will help the management in allocating the resources and selectively invest on the different aspects of the project where the outcome has greater scope.
- **Operational Feasibility –** The study involves recognizing and streamlining the operations in order to make them function seamlessly. The experienced healthcare consultants study the operational feasibility as on how effectively the operations could be structured. This plan includes identifying and finding solutions for the bottlenecks, loopholes, functional gaps and, internal and external communication challenges.
- **Technical Feasibility –** Focusing on the development of the technical aspects is imperative for healthcare success. The professionals will help the management in determining whether the technical resources are on par with the industry and the skill-set of employees is capable enough to make full use of the technical support. This also involves evaluating the healthcare equipment that includes medical equipment, hardware, software and all other electronic communication mechanism and listing the requirements from the perspective.
- **Legal Feasibility –** The healthcare practices in India and allover the world are expected to follow their own guidelines that have been provided by their respective government bodies. The healthcare legal experts from the consultancies support the management in legal feasibility study that includes whether the hospital policies are compatible with the local law, the infrastructural and operational aspects are complimenting the guidelines and the medical qualification of the staff in order to provide medical care as per law. Besides, the healthcare consultants also help in getting all the required paperwork done.

4.2 System Architecture



4.3 System Requirements

HARDWARE INTERFACES

- ✓ PROCESSOR - Pentium-IV (or) Higher
- ✓ RAM- 5 12 MB (or) above
- ✓ HARD DISK- 40GB (or) above
- ✓ INPUT DEVICES - Keyboard, Mouse
- ✓ OUTPUT DEVICES- Monitor

SOFTWARE INTERFACES

- ✓ Operating System – WINDOWS 11
- ✓ Database(backend) – MYSQL, PYTHON
- ✓ Frontend - HTML

4.4 Technologies & Tools

PyCharm 2021.3.1

Sublime text 3

MySQL

CHAPTER 5

SYSTEM DESIGN

5.1 UML DIAGRAMS

USE CASE DIAGRAM

Introduction to UML

The Unified Modeling Language (UML) is a standard visual modeling language intended to be used for

- modeling business and similar processes,
- analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

UML can be applied to diverse application domains (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component software development methods and for various implementation platforms (e.g., J2EE, .NET).

UML is a standard modeling language, not a software development process. UML Specification explained that process

- provides guidance as to the order of a team's activities,
- specifies what artifacts should be developed,
- directs the tasks of individual developers and the team as a whole, and
- offers criteria for monitoring and measuring a project's products and activities

UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP).

UML is not complete and it is not completely visual. Given some UML diagram, we can't be sure to understand depicted part or behavior of the system from the diagram alone. Some information could be intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams.

For example, semantics of multiplicity of actors and multiplicity of use cases on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases.

Name of an abstract classifier is shown in italics while final classifier has no specific graphical notation, so there is no way to determine whether classifier is final or not from the diagram.

Types of UML Diagrams

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing, and deployment.

These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams.

Structural UML Diagrams are-

- Class diagram represent system class, attributes and relationships among the classes
- Package diagram represent splitting of a system into logical groupings and dependency among the grouping
- Object diagram represent a complete or partial view of structure of a modeled system
- Component diagram represent how components are split in a software system and dependencies among the components
- Composite Structure Diagram describes internal structure of classes

- Deployment Diagram describes the hardware used in system implementations

Behavioral UML Diagrams are-

- Activity Diagram represents step by step workflow of business and operational components**.**
- Sequence Diagram represents communication between objects in terms of sequence of messages.
- Use case Diagram: describes functionality of a system in terms of actors, goals as use cases and dependencies among the cases.
- State Diagram Represent states and state transitions.
- Collaboration provides an overview and nodes representing communication diagrams

UML Diagrams

Use Case

A use case diagram is a graphic depiction of the interaction among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements.

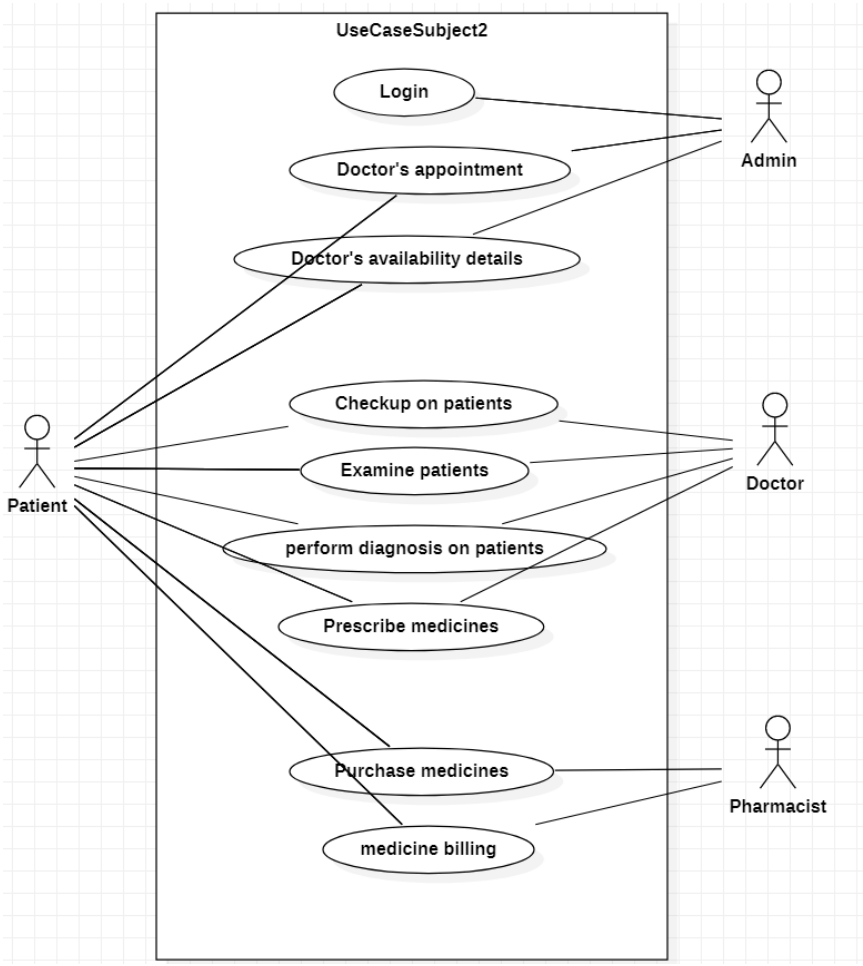


Fig. 5.1 Use case diagram depicting interaction of a patient with admin, doctor and pharmacist

Class Diagrams

Class diagram shows the system's classes, their attributes and operations (or methods), and the relationships among objects.

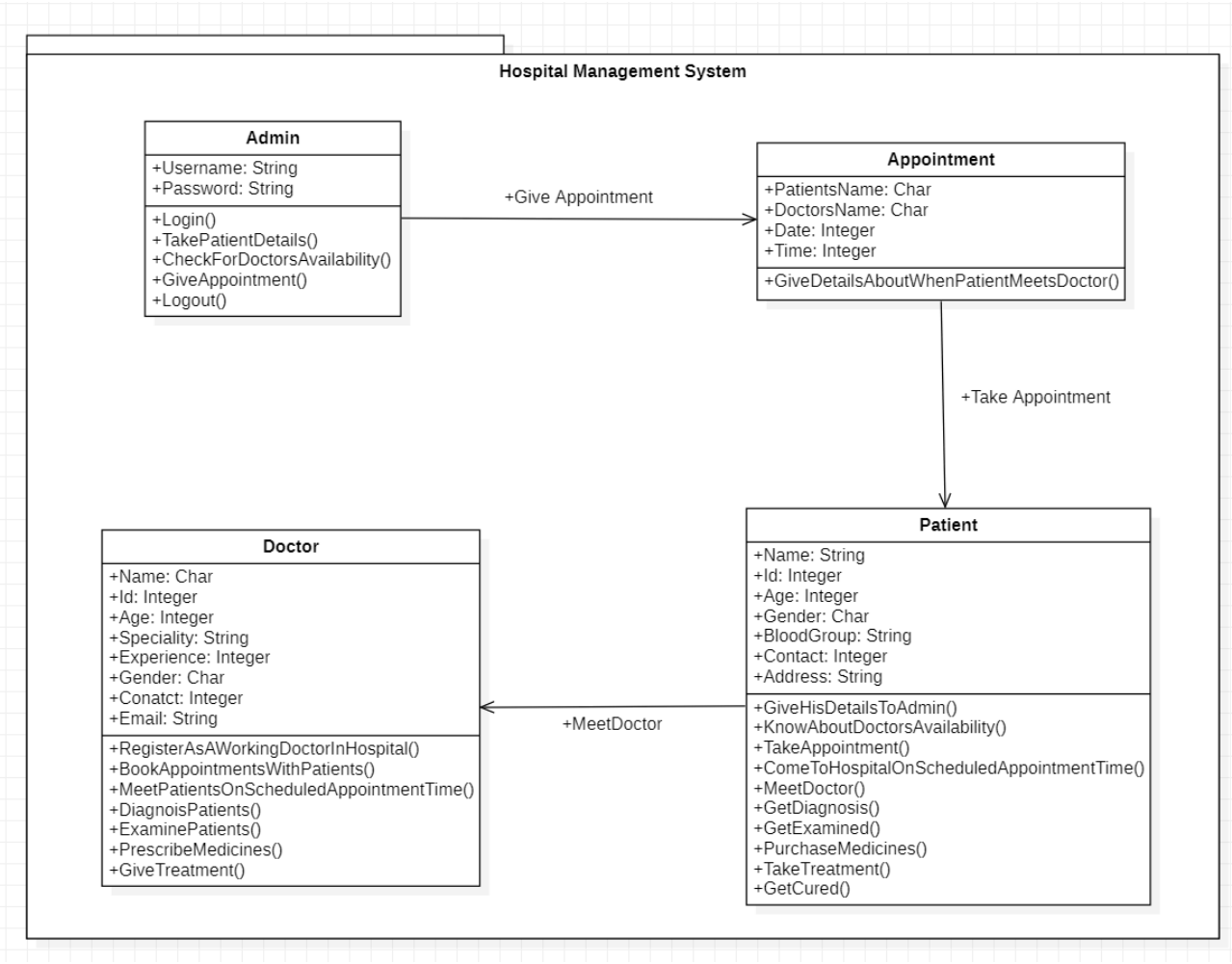


Fig. 5.2 Class Diagram

Sequence Diagram

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.

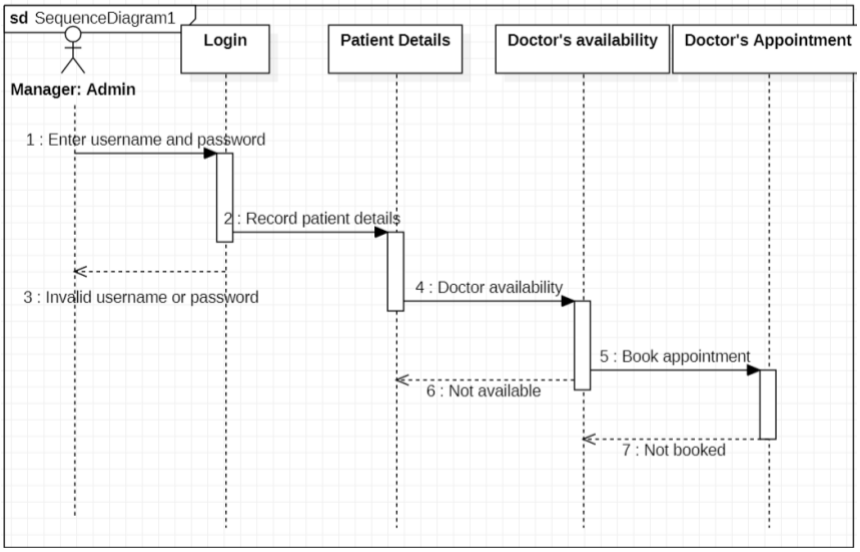


Fig. 5.3 Sequence diagram for Admin

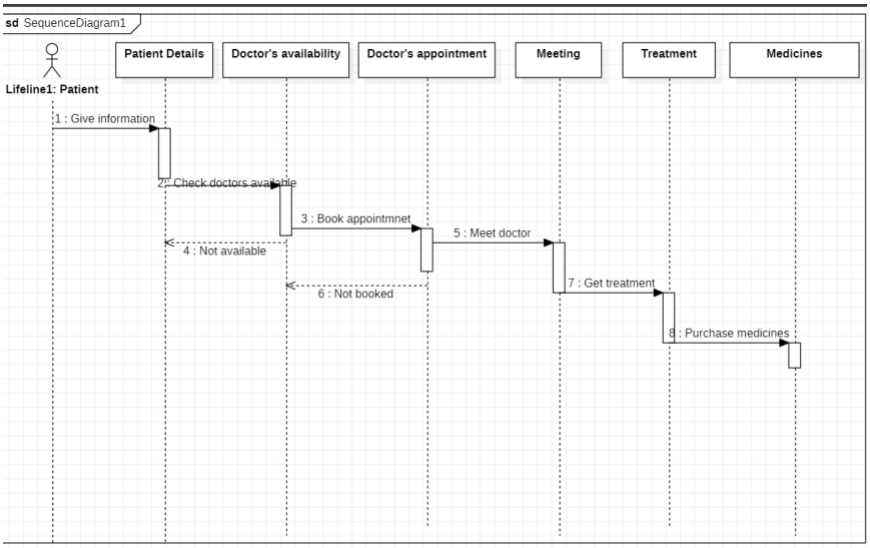


Fig. 5.4 Sequence diagram for patient

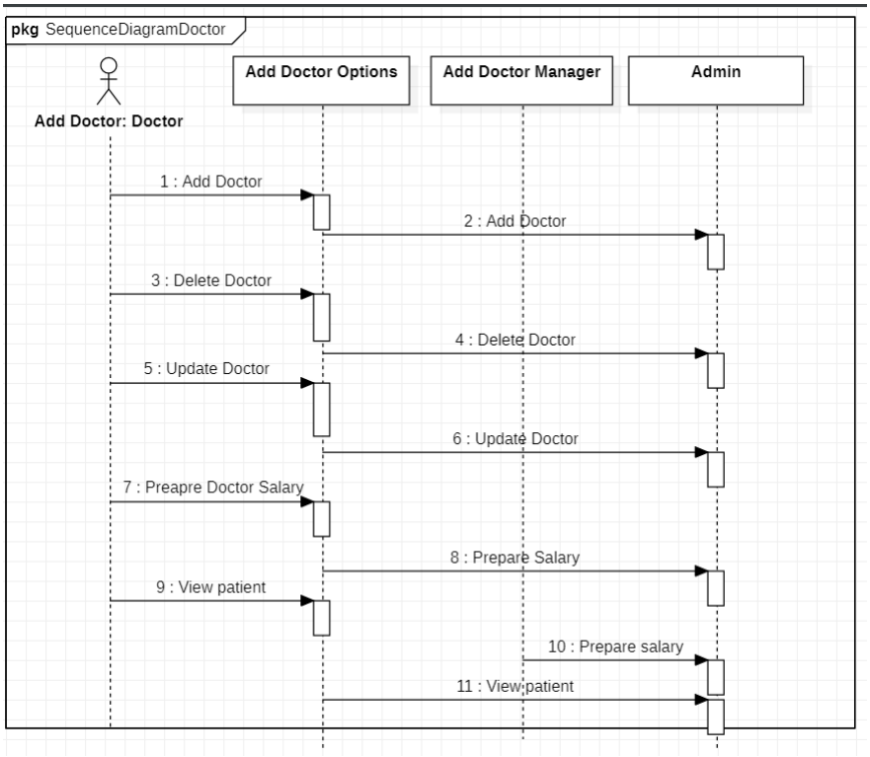


Fig. 5.5 Sequence diagram for doctor

Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

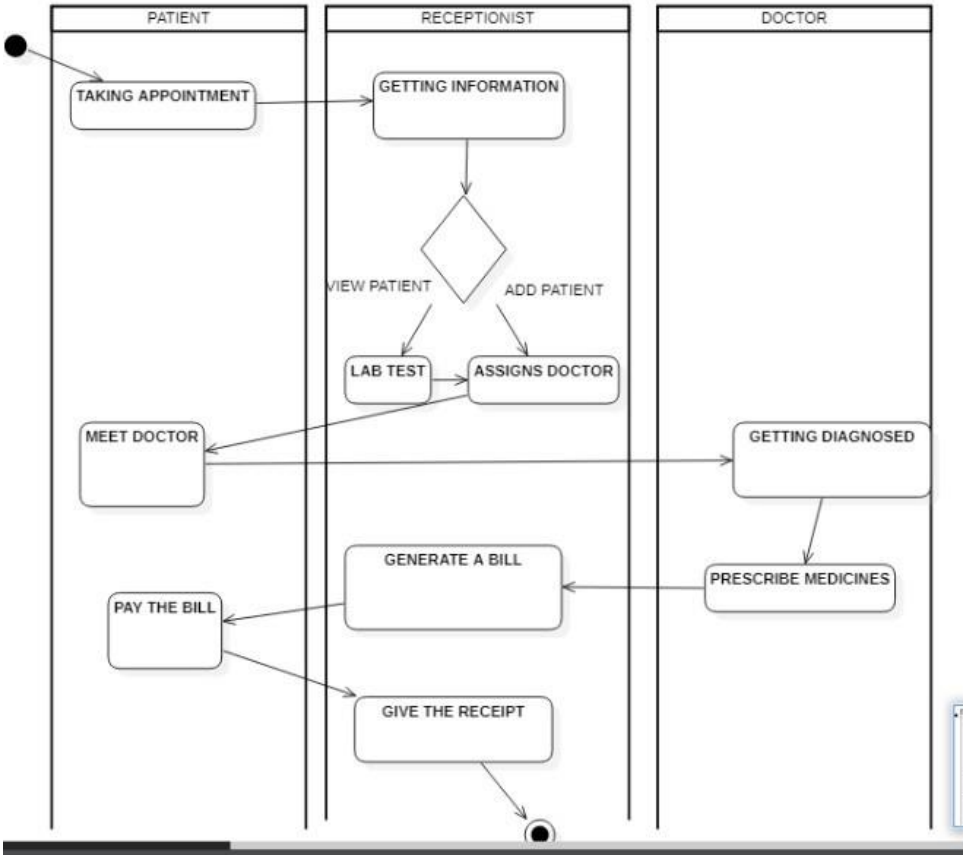


Fig. 5.6 Activity diagram for Care&Cure hospital

5.2 E-R DIAGRAM

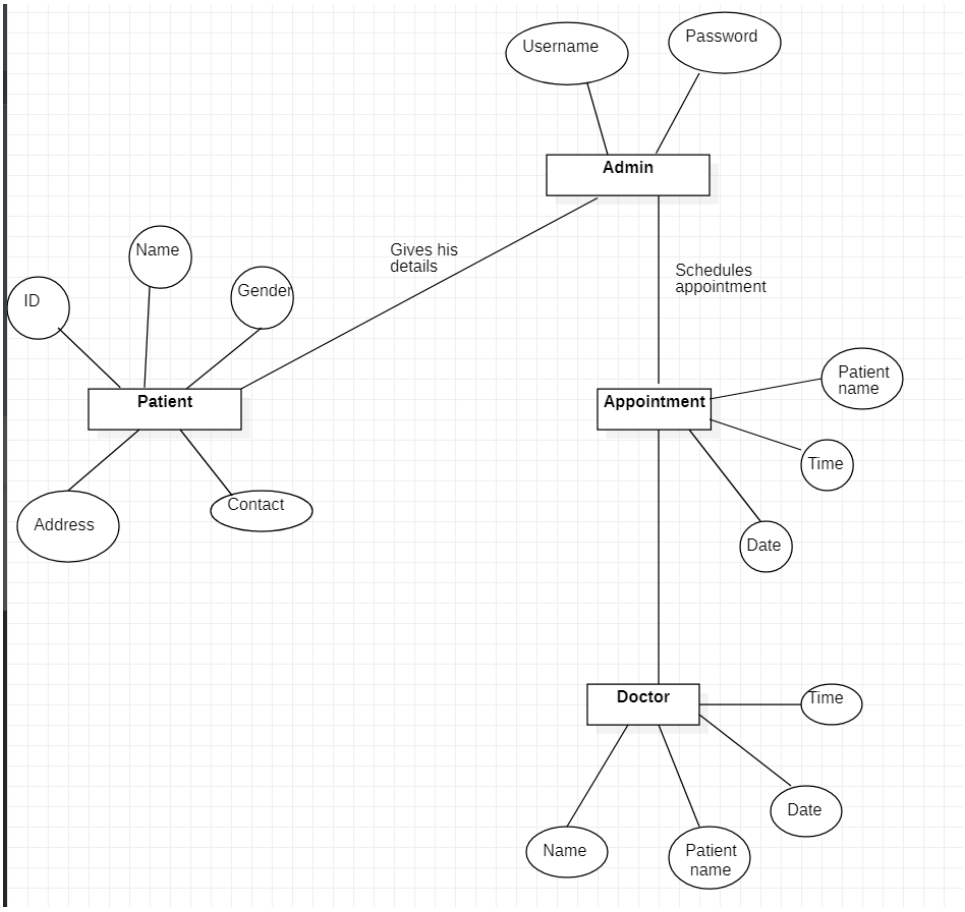


Fig. 5.7 E-R Diagram for Care&Cure hospital

CHAPTER 6

IMPLEMENTATION

6.1 MODULE DESCRIPTION

Login

Admin Login. Any admin available at the hospital can login by entering his username and password. A new user is also allowed to create a new account of his own. After he successfully logged in, he gets access to all data in the hospital and is allowed to update, modify and delete data as required.

Patient Access. Patients that visit the hospital's website can book an appointment with the doctor of their wish at specified date and time. They are also allowed to see the Home page, About page and Contact page.

Home

After a successful login, the home screen appears welcoming the user to the website. A navigation bar is provided where user is allowed to go to other pages of the website.

About us

User gets to see the description and history of hospital on the website

Appointment

Both Admin as well as user can book an appointment with some particular doctor at specified date and time by filling up the 'Add Appointment' form.

Admin, in addition to adding an appointment, also gets to view all the appointments scheduled at the hospital through the 'View Appointment' page.

Doctor

Doctors working at the hospital can register their name, ID and specialty through the ‘Add Doctor’ page and this is visible to all the admins of the website and all the patients that visit the website. All other doctors, patients and admins get to see all the registered doctors through the ‘View Doctor’ page.

Patient

Patients can also register their accounts into the website, as members by filling up the membership form through the ‘Add Patient’ page.

The admin gets to see all the registered patients from the ‘View Patient’ page.

Contact

All visitors of the website can see the contact details of the hospital, including contact no., address and working hours.

Logout

Admins that have logged in to the website can logout if they wish to, but will have to login again if they want to access the website again. It is advised that all the admins logout from the website unless they are using it for security concerns.

6.2 CODE

Filename: login.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Login</title>


<!-- Demo CSS (No need to include it into your project) -->

<link rel="stylesheet" href="css/demo.css">

<style>

:root{

    --success-color: #2ecc71;;

    --error-color: #e74c3c;

}

.container{

    background-color: #fff;

    border-radius: 5px;

    box-shadow: 0 2px 10px rgba(0,0,0,0.3);

    width: 400px;
```

```
    margin: 10px auto;
}
h2{
    text-align: center;
    margin: 0 0 20px;
}
.form{
    padding: 30px 40px;
}
.form-control{
    margin-bottom: 10px;
    padding-bottom: 20px;
    position: relative;
}
.form-control label{
    color:#777;
    display: block;
    margin-bottom: 5px;
}
 .form-control input
{
```

```
    border: 2px solid #f0f0f0;
    border-radius: 4px;
    display: block;
    width: 100%;
    padding: 10px;
    font-size: 14px;
}
.form-control input:focus{
    outline: 0;
    border-color: #777;
}
.form-control.success input {
    border-color: var(--success-color);
}
 .form-control.error input {
    border-color: var(--error-color);
}
.form-control small{
    color: var(--error-color);
    position: absolute;
    bottom: 0;
```

```
    left: 0;
    visibility: hidden;
}
.form-control.error small{
    visibility: visible;
}
.form button {
    cursor: pointer;
    background-color: #3498db;
    border: 2px solid #3498db;
    border-radius: 4px;
    color: #fff;
    display: block;
    padding: 10px;
    font-size: 16px;
    margin-top:20px;
    width:100%;
}
* {
    margin: 0;
    padding: 0;
```

```
    box-sizing: border-box;
}
ul.menu {
    display: inline-block;
    list-style-type: none;
}
li.menu_list {
    height: 85px;
    width: 85px;
    position: relative;
}
li.menu_list .front,
a.side {
    display: flex;
    align-items: center;
    justify-content: center;
    height: 86px;
    width: 100%;
    padding: 30px;
    position: absolute;
    top: 0;
```

```
    left: 0;
    text-decoration: none;
    text-transform: uppercase;
    transition: all .5s ease-out;
    cursor: pointer;
}
li.menu_list .front {
    background-color: #34465d;
    color: #FFFC00;
    transform-origin: 0 0;
}
a.side {
    background-color: #FFFC00;
    color: #34465d;
    transform-origin: 0 0 85px;
    transform: rotateY(-90deg);
}
li.menu_list:hover a.side {
    transform: rotateY(0deg);
}
li.menu_list:hover .front {
```

```
    transform: rotateY(90deg);
}
.right{
    float:right;
}

.left{
    float:left;
}
</style>
  </head>
  <body background="logo.jpg">
  <br/>
  <br/>
  <br/>
  <br/>
  <br/>
<header class="intro">
     <center><h1>Admin Login</h1></center>
     <div class="action"></div>
</header>
```

```
 <main>
<div class="container">
      <form id="form" class="form" action="index.html" method="post">
         <div class="form-control">
            <label for="username">Username</label>
            <input type="text" id="username" placeholder="Enter username">
            <small>Error Message</small>
         </div>
         <div class="form-control">
            <label for="password">Password</label>
            <input type="password" id="password" placeholder="Enter password">
            <small>Error Message</small>
         </div>
         <div class="login">
            <from action="index.html" method="post">
               <div class="form-control">
                  <input type="submit" value="Login" id="login">
               </div>
            </from>
         </div>
      </form>
```

```
</div>
</main>
  <center>
  <h1 class="text-uppercase text-center">Patient access</h1>
        <p>Patients visiting the website can access the website without registering as well,
           <a href="indexp.html">click here to visit</a>
        </p>
  </center>
  </body>
</html>
```

Filename: add_appointment.html

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <script src="https://use.fontawesome.com/3903c9d7fd.js"></script>
   <link       href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.2/css/all.css"
rel="stylesheet">
<link href=""C:\Users\amina\Desktop\Pycharm To Notepad\styles.css.txt"">
   <title>About</title>
</head>
```

```
<style>
:root{
    --success-color: #2ecc71;;
    --error-color: #e74c3c;
}
.container{
    background-color: #fff;
    border-radius: 5px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.3);
    width: 400px;
    margin: 10px auto;
}

h2{
    text-align: center;
    margin: 0 0 20px;
}

.form{
    padding: 30px 40px;
}
```

```
.form-control{
    margin-bottom: 10px;
    padding-bottom: 20px;
    position: relative;
}

.form-control label{
    color:#777;
    display: block;
    margin-bottom: 5px;
}

.form-control input
{
    border: 2px solid #f0f0f0;
    border-radius: 4px;
    display: block;
    width: 100%;
    padding: 10px;
    font-size: 14px;
```

```
}

.form-control input:focus{
    outline: 0;
    border-color: #777;

}

.form-control.success input {
    border-color: var(--success-color);
}

.form-control.error input {
    border-color: var(--error-color);
}

.form-control small{
    color: var(--error-color);
    position: absolute;
    bottom: 0;
    left: 0;
```

```
    visibility: hidden;
}

.form-control.error small{
    visibility: visible;
}
.form button {
    cursor: pointer;
    background-color: #3498db;
    border: 2px solid #3498db;
    border-radius: 4px;
    color: #fff;
    display: block;
    padding: 10px;
    font-size: 16px;
    margin-top:20px;
    width:100%;
}
  * {
    margin: 0;
    padding: 0;
```

```
    box-sizing: border-box;
}
ul.menu {
    display: inline-block;
    list-style-type: none;

}
li.menu_list {
    height: 85px;
    width: 85px;
    position: relative;
}
li.menu_list .front,
a.side {
    display: flex;
    align-items: center;
    justify-content: center;
    height: 86px;
    width: 100%;
    padding: 30px;
    position: absolute;
```

```
    top: 0;
    left: 0;
    text-decoration: none;
    text-transform: uppercase;
    transition: all .5s ease-out;
    cursor: pointer;
}
li.menu_list .front {
    background-color: #34465d;
    color: #FFFC00;
    transform-origin: 0 0;
}
a.side {
    background-color: #FFFC00;
    color: #34465d;
    transform-origin: 0 0 85px;
    transform: rotateY(-90deg);
}
li.menu_list:hover a.side {
    transform: rotateY(0deg);
}
```

```
li.menu_list:hover .front {
    transform: rotateY(90deg);
}
.right{
    float:right;
}

.left{
    float:left;
}
</style>
 <body>
<span class="left">
    <nav class="sidebar">
<ul class="menu">
    <li class="menu_list">
        <span class="front fas fa-home"></span>
        <a href="index.html" class="side">home</a>
    </li>
    <li class="menu_list">
        <span class="front fas fa-info"></span>
```

```
    <a href="about.html" class="side">about</a>
</li>
<li class="menu_list">
    <span class="front fas fa-clock"></span>
    <a href="add_appointment.html" class="side">Appointments</a>
</li>
<li class="menu_list">
    <span class="front fas fa-male"></span>
    <a href="add_doctor.html" class="side">Doctors</a>
</li>
<li class="menu_list">
    <span class="front fab fa-accessible-icon"></span>
    <a href="add_patient.html" class="side">Patients</a>
</li>
    <li class="menu_list">
    <span class="front fas fa-paper-plane"></span>
    <a href="contact.html" class="side">contact</a>
</li>
<li class="menu_list">
    <span class="front fas fa-sign-out-alt"></span>
    <a href="login.html" class="side">Logout</a>
```

```
    </li>
</ul>
</nav>
</span>
<span class="center">
    <main><br/><br/><br/><br/><br/>
      <div class="container">
          <form id="form" class="form" name="myform1" action="appointment.py" method="GET">
              <div class="from-control">
              <h1>         Add Appointment</h1>
              <div class="form-control">
                 <label for="username">Name</label>
                 <input type="text" id="name" name="t1" placeholder="Enter name">
                 <small>Error Message</small>
              </div>
              <div class="form-control">
                 <label for="email">Email</label>
                 <input type="text" id="email" name="t2" placeholder="Enter email">
                 <small>Error Message</small>
              </div>
```

```
            <div class="form-control">
                <label for="id">ID</label>
                <input type="text" id="id" name="t3" placeholder="Enter ID">
                <small>Error Message</small>
            </div>
            <div class="form-control">
                <label for="category">Disease</label>
                <input type="text" id="category" name="t4" placeholder=" disease">
                <small>Error Message</small>
            </div>
            <button>Add</button>
        </form>
    </div>
</main>
<center><a
href="file:///D:/HospitalNew/hospital/templates/view_appointment.html">View
Appointments</a></center>
    </span>
</div>
</body>
</html>
```

Filename: view_appiointment.html

```
import mysql.connector

import webbrowser


conn = mysql.connector.connect(user='root, password='Newtonlaw3', host='localhost', database='hospital'


if conn:

  print("Connected successfully")

else:

  print("Connection not established")


select_appointment = " " "SELECT * FROM appointment""""

cursor = conn.cursor()

cursor.execute(select_appointment)

result = cursor.fetchall()

p = []

Tbl = "<tr><td>NAME></td><td><ID</td><td><SPECIALITY</td></tr>"

p.append(tbl)
```

```
for row in result:
   a = “<tr><td>%s</td>”%row[0]
   p.append(a)
   b = “<tr><td>%s</td>”%row[1]
   p.append(b)
   c = “<tr><td>%s</td>”%row[2]
   p.append(c)
   d = “<tr><td>%s</td>”%row[3]
   p.append(d)
contents = “““<!DOCTYPE html PUBLIC “-//W3C//DTD HTML 4.01 Transitional//EN”>
<html>
<head>
<meta content = “text/html; charset=ISO-8859-1" http-equiv=”content-type”>
<title>View Appointments</title>
</head>
<body>
<table>
%s
</table>
</body>
</html>
```

```
”””” %(p)

filename=’view_appointment.html”


def main(contents, filename):

  output = open(filename, ‘w’)

  output.write(contents)

  output.close()

main(contents, filename)

webbrowser.open(filename)


if conn.is_connected():

  cursor.close()

  conn.close

  print(“mysql connection is closed”)
```

CHAPTER 7

TESTING STRATERGIES AND RESULTS

7.1 TEST CASES

Software testing is performed to verify that the completed software package functions according to the expectations defined by the requirements/specifications. The overall objective to not to find every software bug that exists, but to uncover situations that could negatively impact the customer, usability and/or maintainability.

From the module level to the application level, this article defines the different types of testing. Depending upon the purpose for testing and the software requirements/specs, a combination of testing methodologies is applied. One of the most overlooked areas of testing is regression testing and fault tolerant testing.

In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environment, and
- achieves the general result its stakeholder's desire

Test Case for the scenario: Admin Login

Step1) A simple test case to explain the scenario would be

Test case #	Test case description
1.	Check response when valid username and password is entered

Step 2) In order to execute the test case, you would need to test data, adding it below

Test case #	Test case description	Test data
1.	Check response when valid username and password is entered	Username: admin Password: 123

Step 3) In order to test the test case, a tester needs to perform a specific set of actions on the AUT.

Test case #	Test case description	Test steps	Test data
1.	Check response when valid username and password is entered	1. Enter username 2. Enter password 3. Click login	Username: admin Password: 123

Many times, the Test Steps are not simple as above, hence they need documentation. Also, the author of the test case may leave the organization or go on a vacation or is sick and off duty or is very busy with other critical tasks. A recently hire may be asked to execute the test case. Documented steps will help him and also facilitate reviews by other stakeholders.

Step 4) The goal of test cases in software testing is to check behavior of the AUT for an expected result. This needs to be documented as below

Test case #	Test case description	Test data	Expected result
1.	Check response when valid username and password is entered	Username: admin Password: 123	Login should be successful

During test execution time, the tester will check expected results against actual results and assign a pass or fail status

Test case #	Test case description	Test data	Expected result	Actual result	Pass/Fail
1.	Check response when valid username and password is entered	Username: admin Password: 123	Login should be successful	Login was successful	Pass

Step 5) That apart your test case -may have a field like, Pre - Condition which specifies things that must in place before the test can run. For our test case, a precondition would be to have a browser installed to have access to the site under test. A test case may also include post-Conditions which specifies anything that applies after the test case completes. For our test case, a post condition would be time & date of login is stored in the database.

Test Case for the scenario: Add Appointment

Step1) A simple test case to explain the scenario would be

Test case #	Test case description
1.	Check response when Patient's name entered is Character type, Doctor's name entered is Character type, Date is Integer type and time is Integer type

Step 2) In order to execute the test case, you would need to test data, adding it below

Test case #	Test case description	Test data
1.	Check response when Patient's name entered is Character type, Doctor's name entered is Character type, Date is Integer type and time is Integer type	Patient's name: Alison Doctor's name: Dr. Shawn Date: 12-01-22 Time: 10:00:00

Step 3) In order to test the test case, a tester needs to perform a specific set of actions on the AUT.

Test case #	Test case description	Test steps	Test data
1.	Check response when Patient's name entered is Character type, Doctor's name entered is Character type, Date is Integer type and time is Integer type	1. Enter Patient's name 2. Enter Doctor's name 3. Enter date 4. Enter time	Patient's name: Alison Doctor's name: Dr. Shawn Date: 12-01-22 Time: 10:00:00

Many times, the Test Steps are not simple as above, hence they need documentation.

Also, the author of the test case may leave the organization or go on a vacation or is

sick and off duty or is very busy with other critical tasks. A recently hire may be asked

to execute the test case. Documented steps will help him and also facilitate reviews by other stakeholders.

Step 4) The goal of test cases in software testing is to check behavior of the AUT for an expected result. This needs to be documented as below

Test case #	Test case description	Test data	Expected result
1.	Check response when Doctor's name entered is Character type, Patient's name entered is Character type, Date is Integer type and time is Integer type	Doctor's name: Dr. Shawn Patient's name: Alison Date: 12-01-22 Time: 10:00:00	Appointment should be booked successfully

During test execution time, the tester will check expected results against actual results and assign a pass or fail status

Test case #	Test case description	Test data	Expected result	Actual result	Pass/Fail
1.	Check response when Doctor's name entered is Character type, Patient's name entered is Character type, Date is Integer type and time is Integer type	Doctor's name: Dr. Shawn Patient's name: Alison Date: 12-01-22 Time: 10:00:00	Appointment should be booked successfully	Appointment booked successfully	Pass

Step 5) That apart your test case -may have a field like, Pre - Condition which specifies things that must in place before the test can run. For our test case, a precondition

would be to have a browser installed to have access to the site under test. A test case may also include post-Conditions which specifies anything that applies after the test case completes. For our test case, a post condition would be time & date of login is stored in the database.

Test Case for the scenario: Add Doctor

Step1) A simple test case to explain the scenario would be

Test case #	Test case description
1.	Check response when Name entered is Character type, Email entered is String type, Id is Integer type and Specialty is String type

Step 2) In order to execute the test case, you would need to test data, adding it below

Test case #	Test case description	Test data
1.	Check response when Name entered is Character type, Email entered is String type, Id is Integer type and Specialty is String type	Name: Dr. Shawn Email: Shawn@gmai.com Id: 202 Specialty: Cardiologist

Step 3) In order to test the test case, a tester needs to perform a specific set of actions on the AUT.

Test case #	Test case description	Test steps	Test data
1.	Check response when Name entered is Character type, Email entered is String type, Id is Integer type and Specialty is String type	1. Enter Name 2. Enter Email 3. Enter Id 4. Enter Specialty	Name: Dr. Shawn Email: Shawn@gmai.com Id: 202 Specialty: Cardiologist

Many times, the Test Steps are not simple as above, hence they need documentation. Also, the author of the test case may leave the organization or go on a vacation or is sick and off duty or is very busy with other critical tasks. A recently hire may be asked to execute the test case. Documented steps will help him and also facilitate reviews by other stakeholders.

Step 4) The goal of test cases in software testing is to check behavior of the AUT for an expected result. This needs to be documented as below

Test case #	Test case description	Test data	Expected result
1.	Check response when Name entered is Character type, Email entered is String type, Id is Integer type and Specialty is String type	Name: Dr. Shawn Email: Shawn@gmai.com Id: 202 Specialty: Cardiologist	Doctor should be added successfully

During test execution time, the tester will check expected results against actual results and assign a pass or fail status

Test case #	Test case description	Test data	Expected result	Actual result	Pass/Fail
1.	Check response when Name entered is Character type, Email entered is String type, Id is Integer type and Specialty is String type	Name: Dr. Shawn Email: Shawn@gmai.com Id: 202 Specialty: Cardiologist	Doctor should be added successfully	Doctor added successfully	Pass

Step 5) That apart your test case -may have a field like, Pre - Condition which specifies things that must in place before the test can run. For our test case, a precondition would be to have a browser installed to have access to the site under test. A test case may also include post-Conditions which specifies anything that applies after the test case completes. For our test case, a post condition would be time & date of login is stored in the database.

Test Case for the scenario: Add Patient

Step1) A simple test case to explain the scenario would be

Test case #	Test case description
1.	Check response when Name entered is Character type, Email entered is String type and Id is Integer type

Step 2) In order to execute the test case, you would need to test data, adding it below

Test case #	Test case description	Test data
1.	Check response when Name entered is Character type, Email entered is String type and Id is Integer type	Name: Alison Email: alison@gmail.com Id: 401

Step 3) In order to test the test case, a tester needs to perform a specific set of actions on the AUT.

Test case #	Test case description	Test steps	Test data
1.	Check response when Name entered is Character type, Email entered is String type and Id is Integer type	1. Enter Name 2. Enter Email 3. Enter Id	Name: Alison Email: alison@gmail.com Id: 401

Many times, the Test Steps are not simple as above, hence they need documentation. Also, the author of the test case may leave the organization or go on a vacation or is sick and off duty or is very busy with other critical tasks. A recently hire may be asked to execute the test case. Documented steps will help him and also facilitate reviews by other stakeholders.

Step 4) The goal of test cases in software testing is to check behavior of the AUT for an expected result. This needs to be documented as below

Test case #	Test case description	Test data	Expected result
1.	Check response when Name entered is Character type, Email entered is String type and Id is Integer type	Name: Alison Email: alison@gmail.com Id: 401	Patient should be added successfully

During test execution time, the tester will check expected results against actual results and assign a pass or fail status

Test case #	Test case description	Test data	Expected result	Actual result	Pass/Fail
1.	Check response when Name entered is Character type, Email entered is String type and Id is Integer type	Name: Alison Email: alison@gmail.com Id: 401 Doctor's name: Dr. Shawn Patient's name: Alison Date: 12-01-22 Time: 10:00:00	Patient should be added successfully	Patient added successfully	Pass

Step 5) That apart your test case -may have a field like, Pre - Condition which specifies things that must in place before the test can run. For our test case, a precondition would be to have a browser installed to have access to the site under test. A test case may also include post-Conditions which specifies anything that applies after the test case completes. For our test case, a post condition would be time & date of login is stored in the database.

Types of Testing

White-Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g., in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and systems levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include

- API testing– testing of the application using public and private APIs (application programming interfaces)
- Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies
- Mutation testing methods
- Static testing methods

Black-Box Testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as we

7.2 SCREENSHOTS

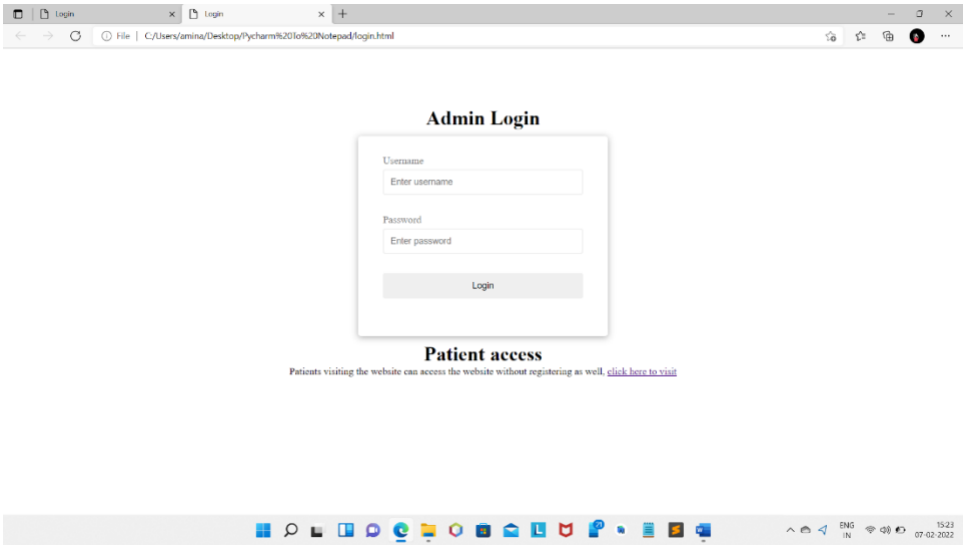


Fig. 7.1 Login page(admin)

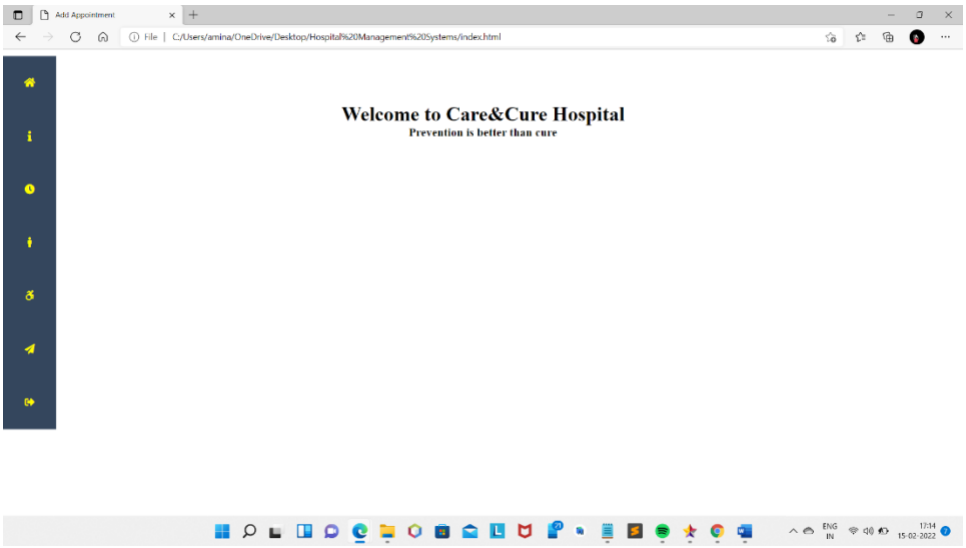


Fig. 7.2 Home page(admin)

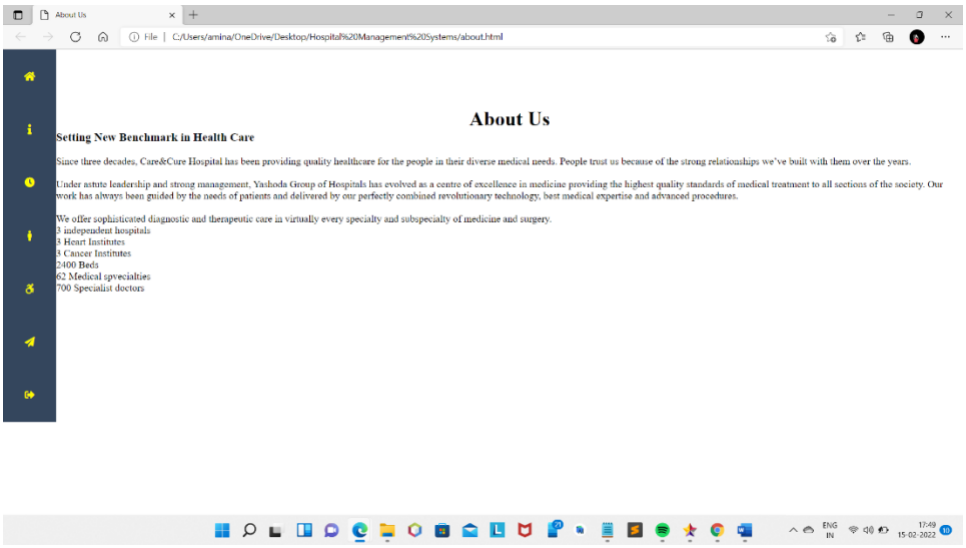


Fig. 7.3 About us page(admin)

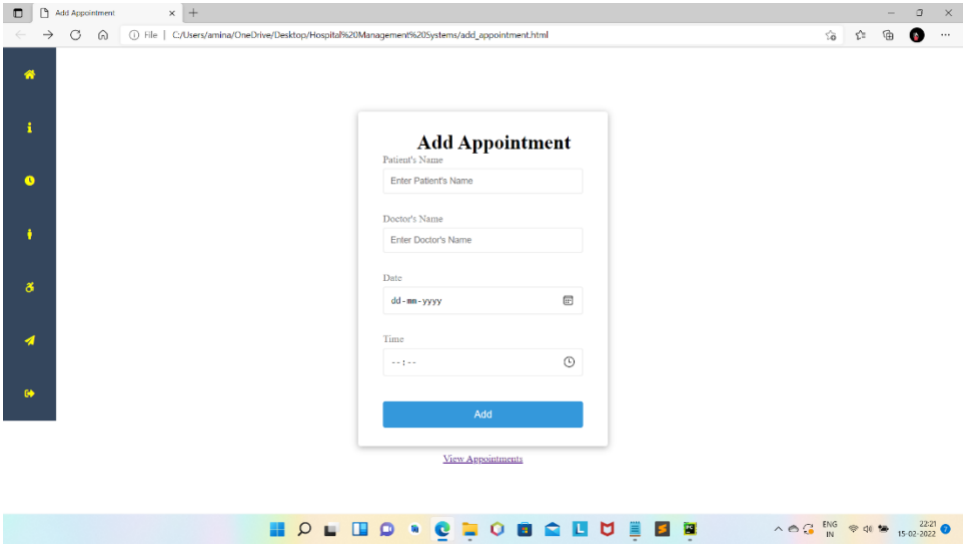


Fig. 7.4 Add Appointment page(admin)

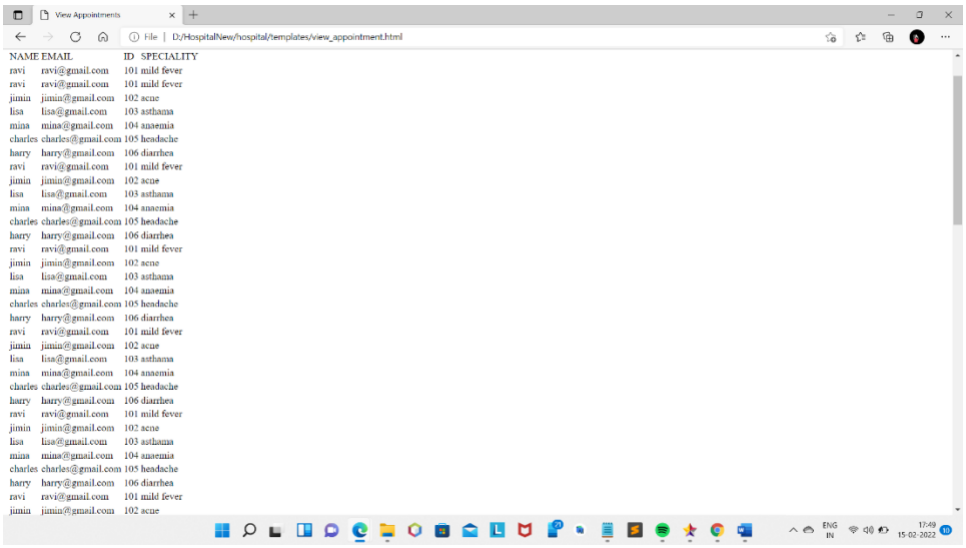


Fig 7.5 View Appointment page(admin)

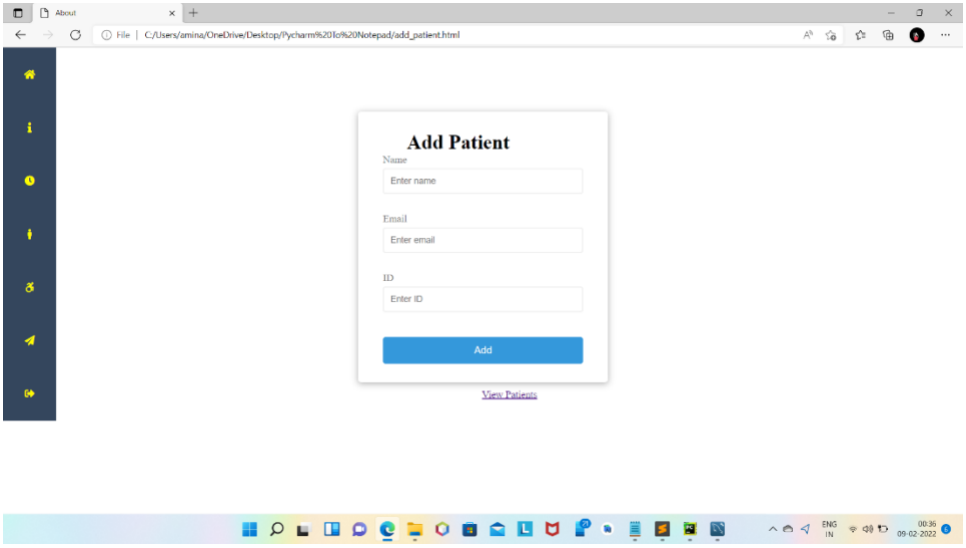


Fig. 7.6 Add patient page(admin)

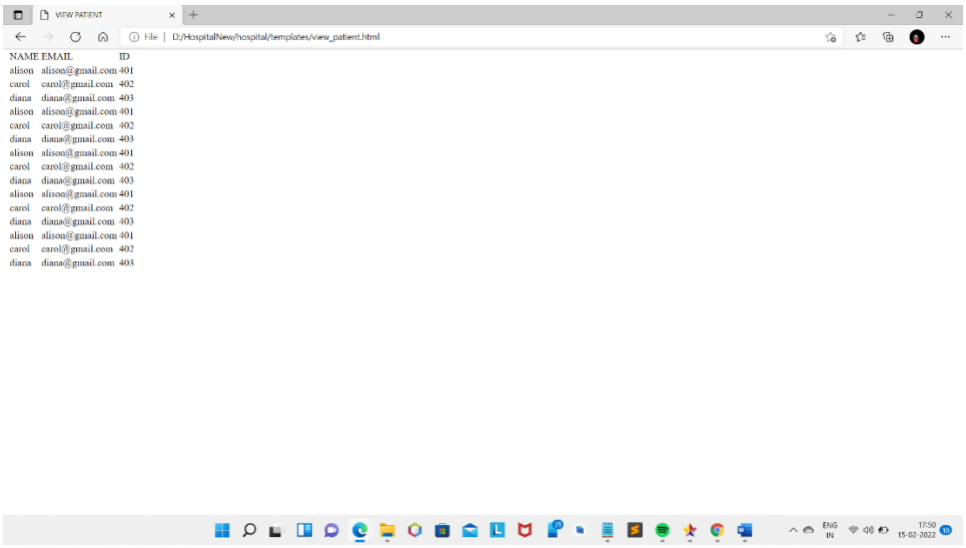


Fig. 7.7 View patient page(admin)

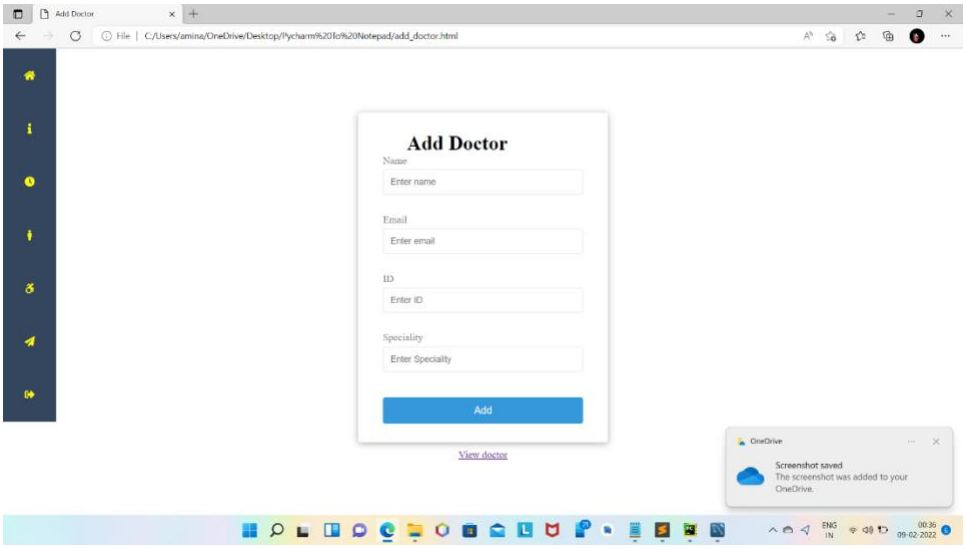


Fig. 7.8 Add Doctor page(admin)

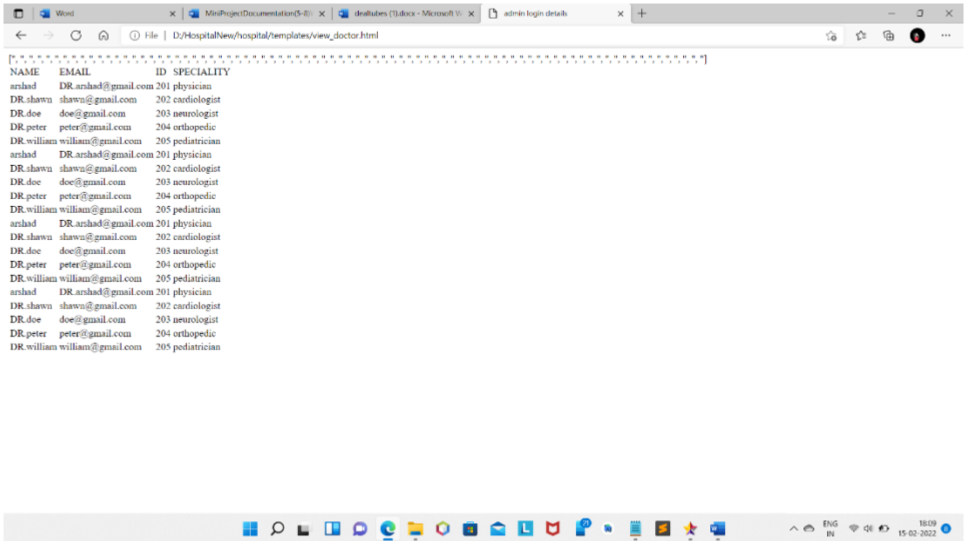


Fig. 7.9 View doctor page(admin)

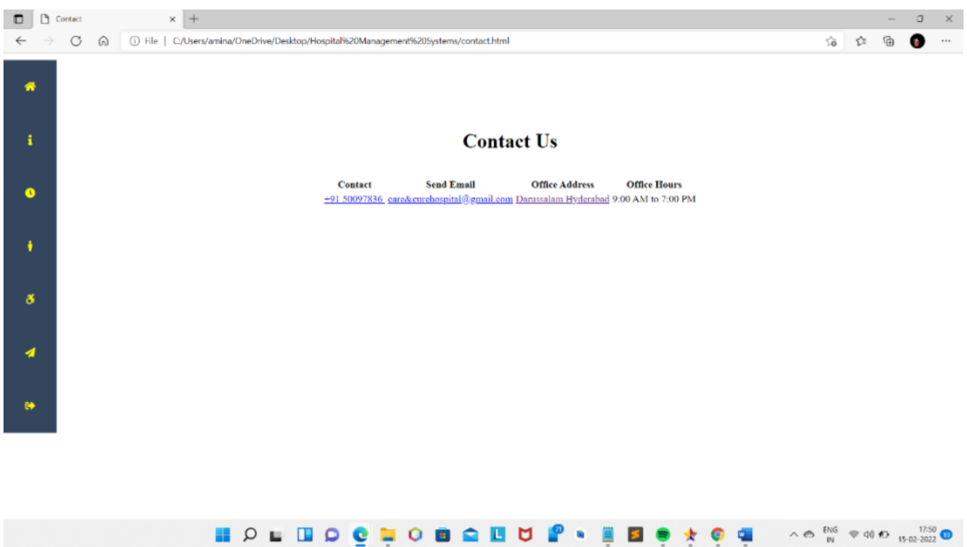


Fig 7.10Contact page page(admin)

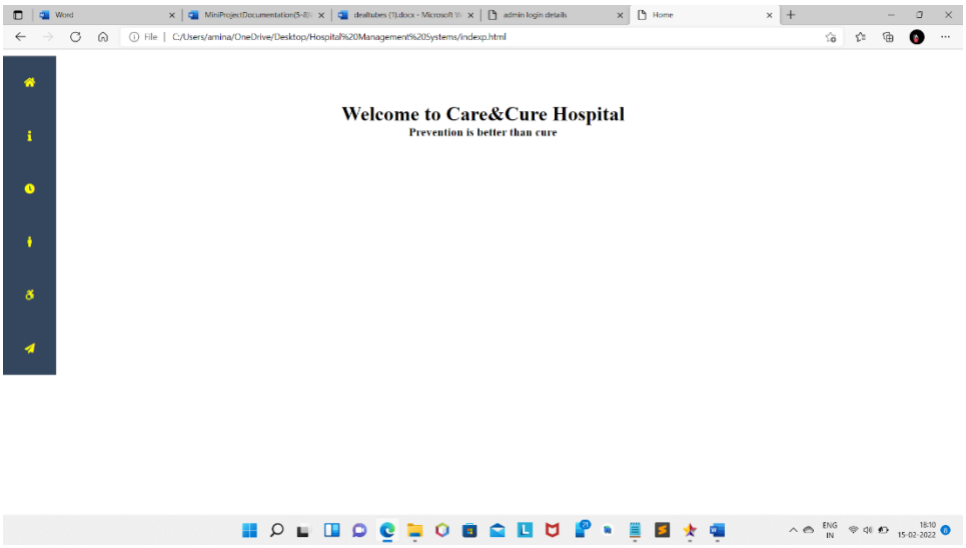


Fig 7.11 Home page (for patients' access)

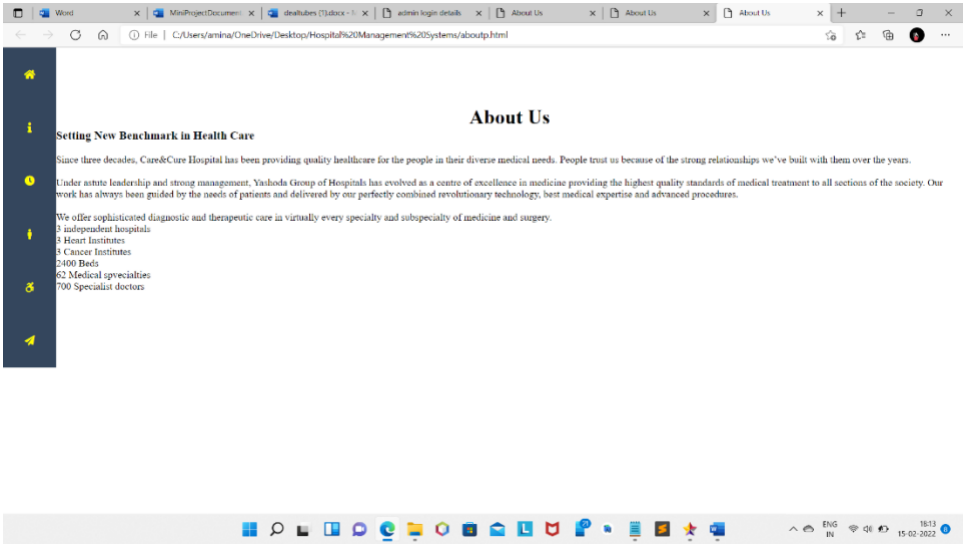


Fig. 7.12 About us page (for patients' access)

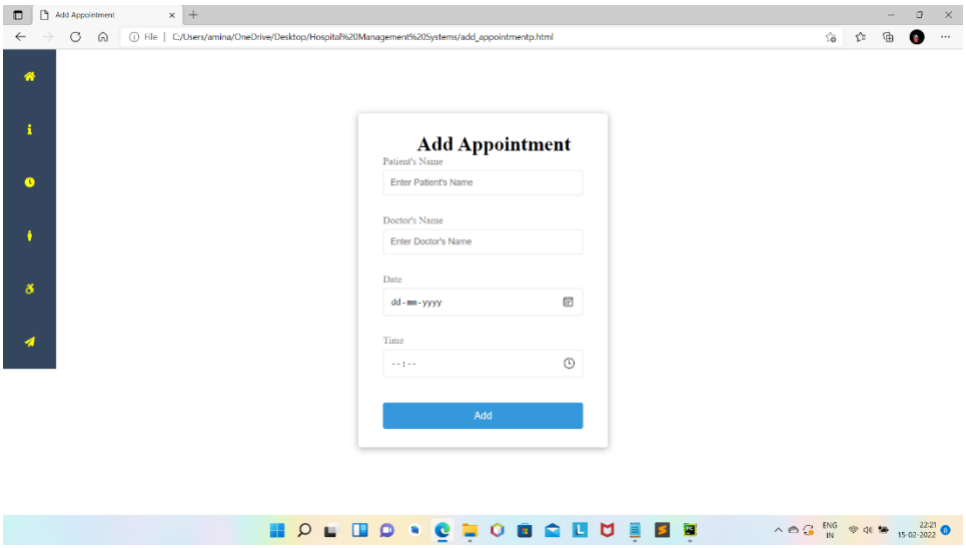


Fig. 7.13 Add Appointment page (for patients' access)

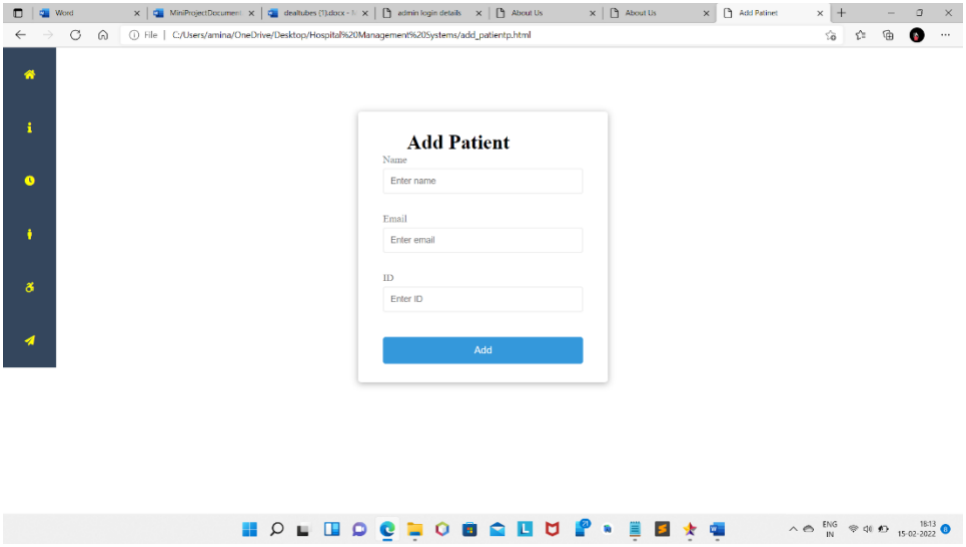


Fig. 7.14 Add patient page (for patients' access)

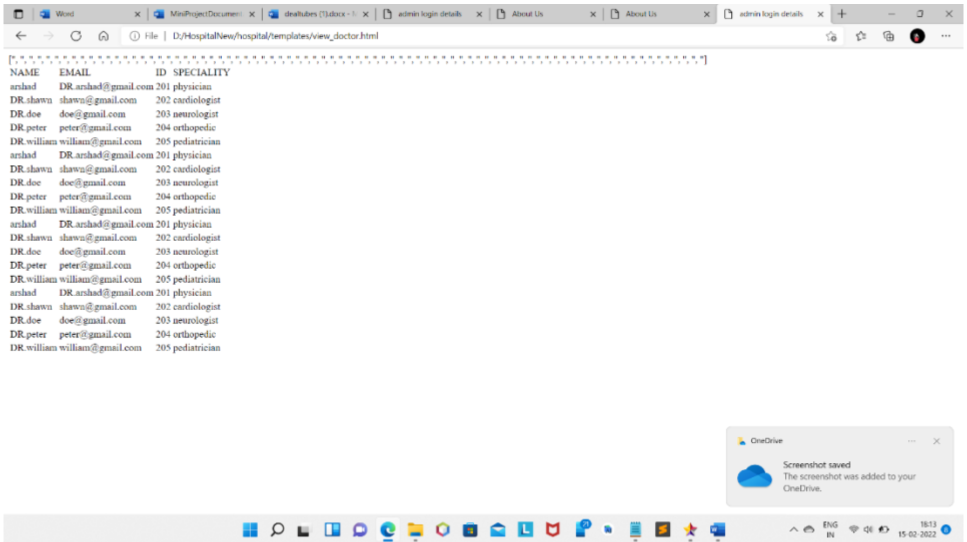


Fig. 7.15 View Doctor page (for patients' access)

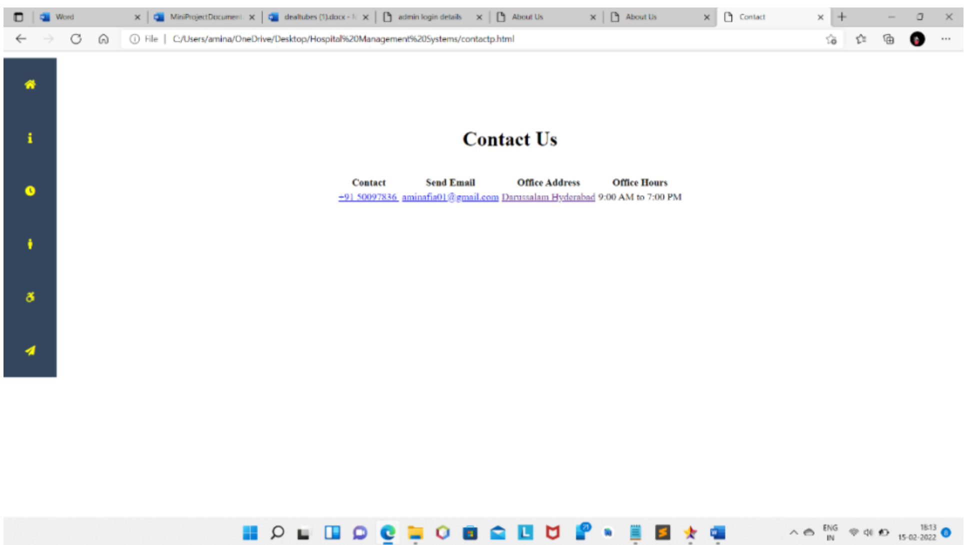


Fig 7.16 Contact page (for patients' access)

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

The project Care&Cure hospital (HMS) is for computerizing the working in a hospital. It is a great improvement over the manual system. The computerization of the system has speed up the process. In the current system, the front office managing is very slow. The hospital managing system was thoroughly checked and tested with dummy data and thus is found to be very reliable. The software takes care of all the requirements of an average hospital and is capable to provide easy and effective storage of information related to patients that come up to the hospital.

It generates test reports and also provides the facility for searching the details of the patient. It also provides billing facility on the basis of patient's status whether it is an indoor or outdoor patient. The system also provides the facility of backup as per the requirement.

FUTURE ENHANCEMENTS

The proposed system is Care&Cure hospital. We can enhance this system by including more facilities like pharmacy system for the stock details of medicines in the pharmacy. Providing such features enable the users to include more comments into the system.

LIMITATIONS

1. The size of the database increases day-by-day, increasing the load on the database back up and data maintenance activity.
2. Training for simple computer operations is necessary for the users working on the system.

CHAPTER 9

REFERENCES

1. Programming the World Wide Web Seventh Edition by Robert W. Sebesta
2. https://www.geeksforgeeks.org/
3. https://www.w3schools.com/