



valtech_

HEALTH CARE EXPRESS

Table of contents

Chapter 1: INTRODUCTION 1.1 Introduction 1.2 Problem Statement 1.3 Purpose of the Project 1.4 Objective 1.5 Scope of the Project	1
Chapter 2: OVERALL DESCRIPTION 2.1 Overall Description 2.2 Modules 2.2.1 Admin 2.2.2 Nurse 2.2.3 Doctor 2.3 Product Perspective 2.3.1 User Interface 2.3.2 Hardware Interface 2.3.3 Software Interface 2.4 Use Case Diagram 2.5 Class Diagram 2.6 Sequence Diagram 2.7 ER Diagram	3
Chapter 3: ANALYSIS 3.1 Technical Requirements 3.2 Software Specification 3.2.1 HTML 3.2.2 CSS 3.2.3 Java 3.3 What Spring Boot Can Do 3.3.1 Why Spring Boot 3.3.2 How does It Work 3.3.3 Spring Boot Starters 3.3.4 Auto Configuration	10

Chapter 4: DATABASE MySQL Database 4.1 Table Descriptions 4.2 Database Scripts 4.2.1 USERS Table 4.2.2 PATIENT_DETAILS Table 4.2.3 DOCTOR_COMMENTS Table 4.2.4 HOSPITAL Table 4.2.5 AVAILABILITY Table 4.3 Database Table Designs	19
Chapter 5: SNAPSHOTS	27

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Hospital Management System is an organized computerized system designed and programmed to avoid the death rates of patients while travelling in ambulance to the hospital.

This project Hospital Management system includes registration of patients, storing their details into the system. The software has the facility to give a unique ID for every patient and stores the clinical details of every patient. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or the nurse. Only they can add data into the database. The data can be retrieved easily.

1.2 PROBLEM STATEMENT

Whenever any person is critically, he/she needs to be taken to the hospital through ambulance. While travelling through ambulance the major problem is the doctor's unavailability to attend the patient or monitor situation of the patient.

Due to which, majority of death happens in the ambulances. The system will facilitate the doctors to monitor the health and condition of the patient during the travel between patients' place and hospital. Hence, the life of the patient can be treated as early as possible.

1.3 PURPOSE OF THE PROJECT

The purpose of this Documentation is to outline both the functional and non-functional requirements of the subject "Health Care Express". In addition to said requirements, the document also provides a detailed profile of the external interfaces, performance considerations imposed on the subsequent implementation. It is the intention that the presented set of requirements possesses the following qualities: correctness, unambiguousness, completeness, consistency, verifiability, modifiability and traceability. Consequently, the document should act as a foundation for efficient and well-managed project completion and further serve as an accurate reference in the future. It will not only provide an extensive capacity for project planning and progress assessment, but it will further assist with senior developers/SME groups interactions.

1.4 OBJECTIVE

- Recording the information about the patient that gets registered by the nurse.
- This information can be viewed by the doctor, which helps him advise about the treatment and instructions to be followed by the nurse.
- This instruction provided by the doctor is sent to the staff, in the ambulance, so that patient is treated accordingly.

1.5 SCOPE OF THE PROJECT

The patient details are made available to the staff/nurse and the doctor, as well. So that the staff in the ambulance can treat the patient accordingly and update about the situation of the patient to the doctor. This reduces the criticality of the patient which helps to restore health swiftly and safely.

CHAPTER 2

OVERALL DESCRIPTION

2.1 OVERALL DESCRIPTION

The following section presents an overall description of the subject “Health Care Express (HCE)”. In particular, the product has been put into perspective through a detailed assessment of the user, hardware, software and communication interfaces requirements.

2.2 MODULES

The entire project mainly consists of 3 modules, which are

- 1) Admin Module
- 2) Nurse Module
- 3) Doctor Module

➤ ADMIN MODULE

- Admin logs-in with his/her credentials.
- He can register new hospitals.
- Update the hospital’s availability.
- Provides list of hospitals available
- Provides doctor specialty details.

➤ NURSE MODULE

- Doctor logs-in with his/her credentials, if doesn’t have, he should himself get registered, before logging in.
- Nurse registers the patients.
- Manages patient and updates doctor about the patient’s condition.
- Can see list of patients.

➤ DOCTOR MODULE

- Doctor logs-in with his/her credentials, if he doesn't have, he should himself get registered as a new doctor, before logging in.
- When doctor is notified about the patient being registered, he views reports and history of the patient from the nurse and treats accordingly.
- Can see and update patient comments.

2.3 PRODUCT PERSPECTIVE

The software described in this documentation is the software for a complete HCE system. The system merges various hardware and software elements and further interfaces with external systems. Thus, while the software covers the majority of the system's functionality, it relies on several external interfaces for persistence and unhandled tasks, as well as physically interfacing with humans.

2.3.1 USER INTERFACE

There is one separate user interface used by the HCE software, related to an interfaced physical hardware device.

➤ Surface Computer UI

The Surface Computer UI is the interface used by all three users i.e., Admin, Doctor and Nurse. This interface uses the surface computer paradigm.

2.3.2 HARDWARE INTERFACE

There is one external hardware device used by the HCE, each related to a user interface. However, they should be fully capable computers that can use textual data from the server along with local UI/interpretation code to display UI elements and take input. All order and transaction records should be stored on the database server. In all cases, the hardware device takes information from the HCE and processes the information to display. It also provides user input

information to the HCE.

2.3.3 SOFTWARE INTERFACE

The HCE will interface with a Database Management System (DBMS) that stores the information necessary for the HCE to operate. The DBMS must be able to provide, on request and with low latency, data concerning the patient details and doctor availability.

2.4 USE CASE DIAGRAM

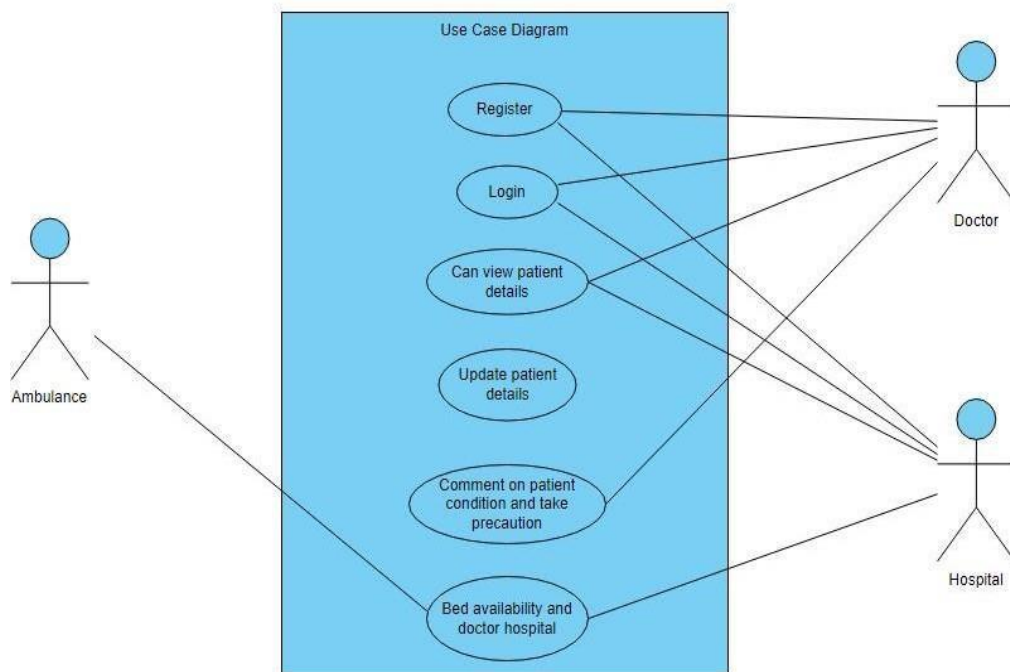


Fig 2.1 Use Case Diagram

2.5 DATABASE DESIGN

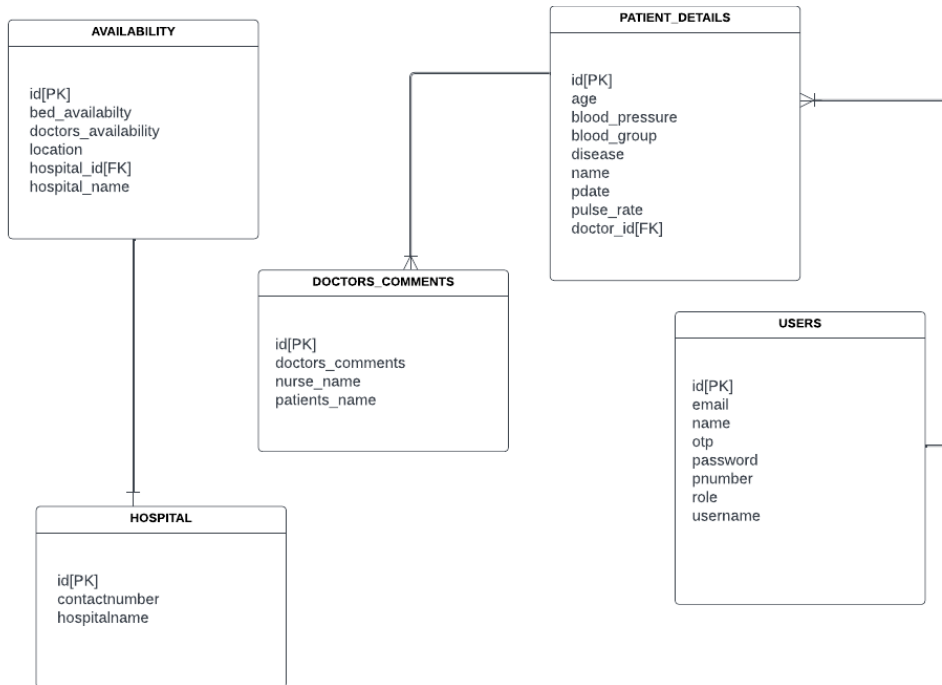


Fig 2.2 Database Design

2.6 SEQUENCE DIAGRAM

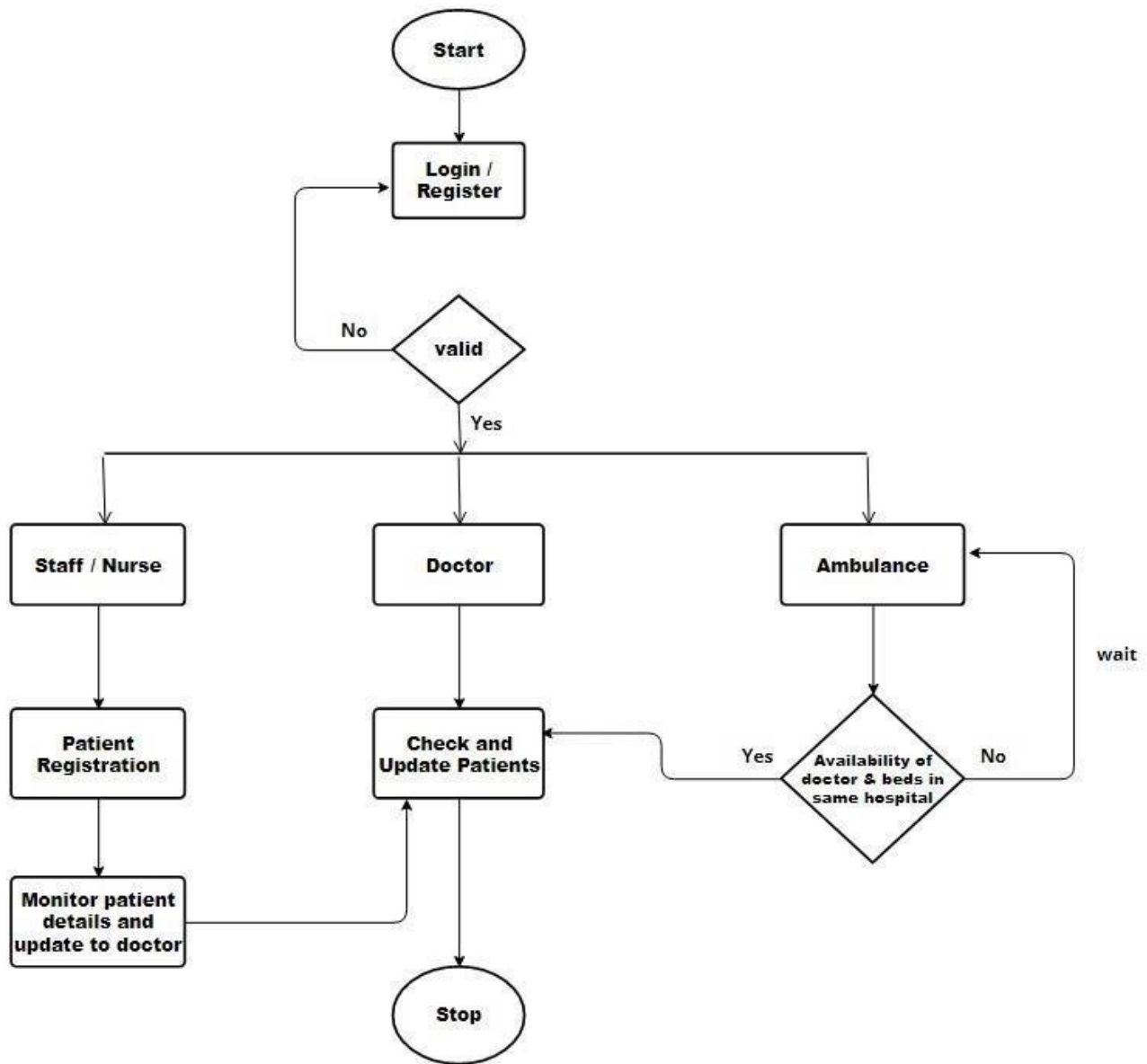


Fig 2.3 Class Diagram

2.7 ER DIAGRAM

ER diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

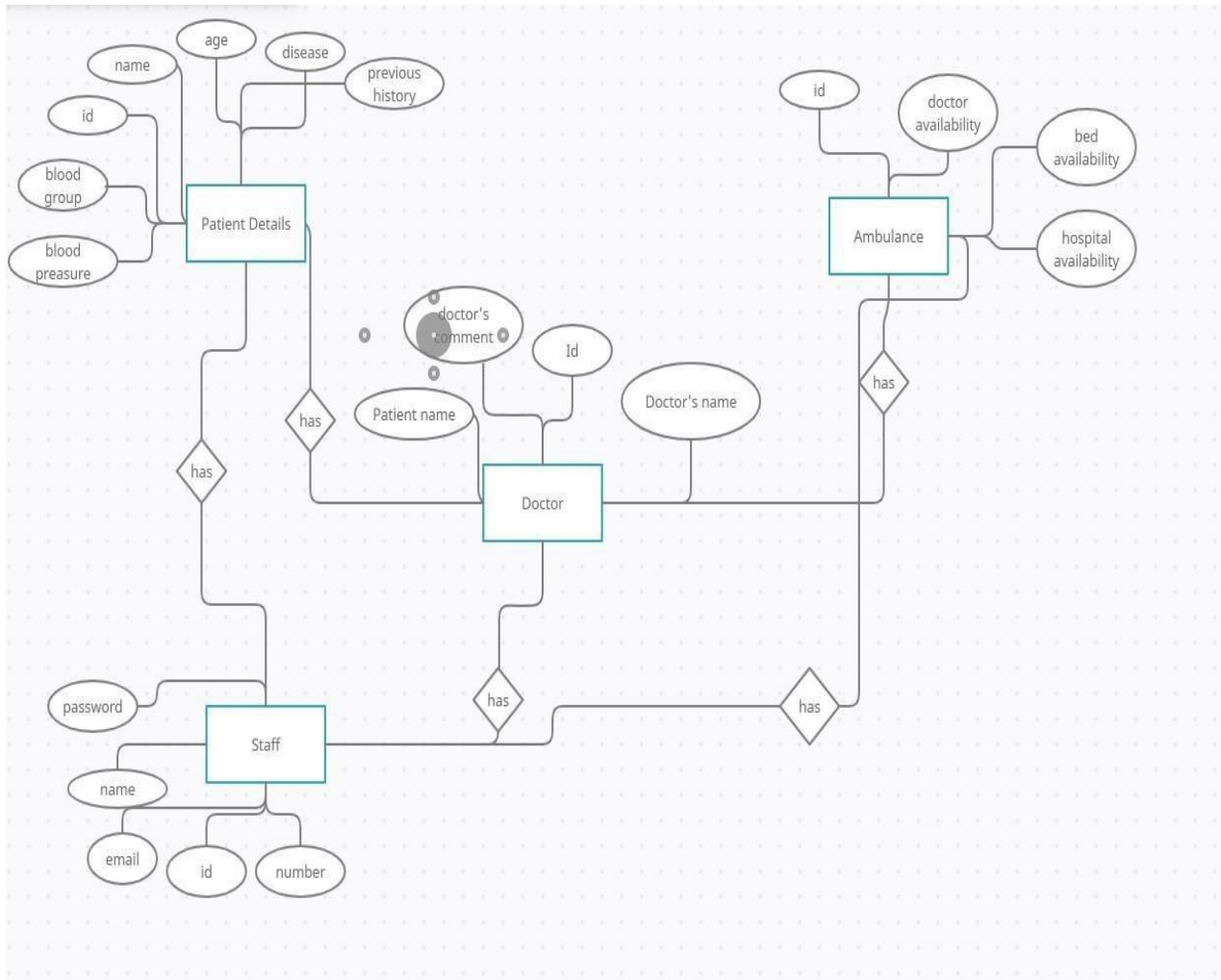


Fig 2.4 ER Diagram

CHAPTER 3

ANALYSIS

3.1 TECHNICAL REQUIREMENTS

- Spring Tool Suite (STS)
- Java
- Oracle Database
- HTML, CSS, JavaScript
- Apache Tom Cat Server
- Maven
- JIRA

Spring Tool Suite:

Spring Tool Suite (STS) is a java IDE tailored for developing Spring-based enterprise applications. It is easier, faster, and more convenient. And most importantly it is based on Eclipse IDE.

Java:

Java is the official language for Android mobile app development.

H2 Database:

H2 is an open-source lightweight Java database. It can be embedded in Java applications or run in the client-server mode.

HTML, CSS, JS:

- HTML is the code that is used to structure a web page and its content
- CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts
- JS is used by programmers across the world to create dynamic and interactive web content like applications and browsers.

Tom Cat server:

Apache Tomcat is a web container. It allows the users to run Servlet and JAVA Server Pages that are based on the web-applications.

Maven:

Maven is used for building, publishing, and deploying several projects.

JIRA:

Jira helps teams plan, assign, track, report, and manage work and brings teams together for everything from agile software development and customer support to start-ups and enterprises.

3.2 SOFTWARE SPECIFICATION

3.2.1 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 come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example . 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.

3.2.2 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.

3.2.3 JAVA

- **Object Oriented** – In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** – Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** – Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure** – With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** – Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable** – Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

3.3 WHAT CAN SPRING BOOT DO?

Spring Boot is an open-source Java-based framework used to create a micro-Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.

3.3.1 WHY SPRING BOOT?

- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides a powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto configured; no manual configurations are needed.
- It offers annotation-based spring application
- Eases dependency management
- It includes Embedded Servlet Container

3.3.2 HOW DOES IT WORK?

Spring Boot automatically configures your application based on the dependencies you have added to the project by using **@EnableAutoConfiguration** annotation.

For example, if MySQL database is on your class path, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.

The entry point of the spring boot application is the class contain **@SpringBootApplication** annotation and the main method.

Spring Boot automatically scans all the components included in the project by using **@ComponentScan** annotation.

3.3.3 SPRING BOOT STARTERS

Handling dependency management is a difficult task for big projects. Spring Boot resolves this problem by providing a set of dependencies for developer's convenience.

For example, if you want to use Spring and JPA for database access, it is sufficient if you include spring-boot-starter-data-jpa dependency in your project.

Examples:

Spring Boot Starter Actuator dependency is used to monitor and manage your application. Its code is shown below

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-actuator</artifactId>
```

```
</dependency>
```

Spring Boot Starter Security dependency is used for Spring Security. Its code is shown below

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-security</artifactId>
```

```
</dependency>
```

Spring Boot Starter web dependency is used to write a Rest Endpoints. Its code is shown below

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-web</artifactId>
```

```
</dependency>
```

Spring Boot Starter Thyme Leaf dependency is used to create a web application. Its code is shown below

```
<dependency>  
  
<groupId>org.springframework.boot</groupId>  
  
<artifactId>spring-boot-starter-thymeleaf</artifactId>  
  
</dependency>
```

Spring Boot Starter Test dependency is used for writing Test cases. Its code is shown below

```
<dependency>  
  
<groupId>org.springframework.boot</groupId>  
  
<artifactId>spring-boot-starter-test</artifactId>  
  
</dependency>
```

3.3.4 AUTO CONFIGURATION

Spring Boot Auto Configuration automatically configures your Spring application based on the JAR dependencies you added in the project. For example, if MySQL database is on your class path, but you have not configured any database connection, then Spring Boot auto configures an in-memory database.

For this purpose, **@EnableAutoConfiguration** annotation or **@SpringBootApplication** annotation must be added to the main class file. By this, the Spring Boot application will be automatically configured.

```
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
```

```
@EnableAutoConfiguration
```

```
public class DemoApplication { public
```

```
static void main(String[] args){
```

```
    SpringApplication.run(DemoApplication.class, args);
```

```
    }
```

```
}
```

3.3.5 SPRINGBOOT APPLICATION

The entry point of the Spring Boot Application is the class contains

@SpringBootApplication annotation. This class should have the main method to run the Spring Boot application. **@SpringBootApplication** annotation includes Auto- Configuration, Component Scan, and Spring Boot Configuration.

If **@SpringBootApplication** annotation to the class, there is no need to add the **@EnableAutoConfiguration**, **@ComponentScan** and **@SpringBootApplication** annotation. The **@SpringBootApplication** annotation includes all other annotations.

```
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
```

```
public class DemoApplication {
```

```
public static void main(String[] args) {
```

```
    SpringApplication.run(DemoApplication.class, args);
```

```
    }
```

```
}
```

3.3.6 COMPONENT SCAN

Spring Boot application scans all the beans and package declarations when the application initializes. The **@ComponentScan** annotation for your class file to scan your components added in your project is needed.

```
import org.springframework.boot.SpringApplication;

import org.springframework.context.annotation.ComponentScan;

@ComponentScan
public class DemoApplication {

    public static void main(String[] args){

        SpringApplication.run(DemoApplication.class, args);

    }
}
```

CHAPTER 4

DATABASE

MySQL DATABASE

SQL stands for Structured Query Language. It's used for relational **databases**. A **SQL database** is a collection of tables that stores a specific set of structured data.

4.1 TABLE DESCRIPTION

➤ AVAILABILITY

COLUMN NAME	DATATYPE
ID[PK]	NUMBER(10,0)
BED_AVAILABILITY	VARCHAR2(255 CHAR)
DOCTORS_AVAILABILITY	VARCHAR2(255 CHAR)
LOCATION	VARCHAR2(255 CHAR)
HOSPITAL_ID[FK]	NUMBER(10,0)
HOSPITAL_NAME	VARCHAR2(255 CHAR)

➤ PATIENT_DETAILS

COLUMN NAME	DATATYPE
ID	NUMBER(10,0)
AGE	NUMBER(10,0)
BLOOD_PRESSURE	VARCHAR2(255 CHAR)
BLOODGROUP	VARCHAR2(255 CHAR)
DISEASE	VARCHAR2(255 CHAR)
NAME	VARCHAR2(255 CHAR)
PDATE	TIMESTAMP(6)
PULSE_RATE	NUMBER(10,0)
DOCTOR_ID	NUMBER(10,0)

➤ USERS

COLUMN NAME	DATATYPE
ID	NUMBER(10,0)
EMAIL	VARCHAR2(255 CHAR)
NAME	VARCHAR2(255 CHAR)
OTP	NUMBER(10,0)
PASSWORD	VARCHAR2(255 CHAR)
PNUMBER	VARCHAR2(255 CHAR)
ROLE	VARCHAR2(255 CHAR)

USERNAME	VARCHAR2(255 CHAR)
ID	NUMBER(10,0)

➤ HOSPITAL

COLUMN NAME	DATATYPE
ID	NUMBER(10,0)
CONTACTNUMBER	VARCHAR2(255 CHAR)
HOSPITALNAME	VARCHAR2(255 CHAR)

➤ DOCTORS_COMMENTS

COLUMN NAME	DATATYPE
ID	NUMBER(10,0)
DOCTORS_COMMENTS	VARCHAR2(255 CHAR)
NURSENAME	VARCHAR2(255 CHAR)
PATIENTS_NAME	VARCHAR2(255 CHAR)

4.2 DATABASE SCRIPTS

4.2.1

a) DDL for USERS Table

```
CREATE TABLE "TEAM4"."USERS"
(
  "ID" NUMBER(10,0),
  "EMAIL" VARCHAR2(255 CHAR),
  "NAME" VARCHAR2(255 CHAR),
  "OTP" NUMBER(10,0),
  "PASSWORD" VARCHAR2(255 CHAR),
  "PNUMBER" VARCHAR2(255 CHAR),
  "ROLE" VARCHAR2(255 CHAR),
  "USERNAME" VARCHAR2(255 CHAR)
)
```

b) Constraints for USERS Table

```
ALTER TABLE "TEAM4"."USERS" MODIFY ("ID" NOT NULL ENABLE);
ALTER TABLE "TEAM4"."USERS" MODIFY ("EMAIL" NOT NULL ENABLE);
ALTER TABLE "TEAM4"."USERS" MODIFY ("USERNAME" NOT NULL ENABLE);
ALTER TABLE "TEAM4"."USERS" ADD PRIMARY KEY ("ID")
ALTER TABLE "TEAM4"."USERS" ADD CONSTRAINT "UK_6DOTKOTT2KJSP8VW4D0M25FB7" UNIQUE ("EMAIL")
ALTER TABLE "TEAM4"."USERS" ADD CONSTRAINT "UK_R43AF9AP4EDM43MMTQ010DDJ6" UNIQUE ("USERNAME")
```

4.2.2

a) DDL for PATIENT_DETAILS Table

```
CREATE TABLE "TEAM4"."PATIENT_DETAILS"
(
  "ID" NUMBER(10,0),
  "AGE" NUMBER(10,0),
  "BLOOD_PRESSURE" VARCHAR2(255 CHAR),
  "BLOODGROUP" VARCHAR2(255 CHAR),
  "DISEASE" VARCHAR2(255 CHAR),
  "NAME" VARCHAR2(255 CHAR),
  "PDATE" TIMESTAMP (6),
  "PULSE_RATE" NUMBER(10,0),
  "DOCTOR_ID" NUMBER(10,0)
)
```

b) Constraints for PATIENT_DETAILS Table

```
ALTER TABLE "TEAM4"."PATIENT_DETAILS" MODIFY ("ID" NOT NULL ENABLE);
ALTER TABLE "TEAM4"."PATIENT_DETAILS" MODIFY ("AGE" NOT NULL ENABLE);
ALTER TABLE "TEAM4"."PATIENT_DETAILS" MODIFY ("NAME" NOT NULL ENABLE);
ALTER TABLE "TEAM4"."PATIENT_DETAILS" MODIFY ("PULSE_RATE" NOT NULL ENABLE);
ALTER TABLE "TEAM4"."PATIENT_DETAILS" ADD PRIMARY KEY ("ID")
```

c) Ref Constraints for Table PATIENT_DETAILS

```
ALTER TABLE "TEAM4"."PATIENT_DETAILS" ADD CONSTRAINT "FK14QQCDDRG4PX9F0AEQ6KUB9TN" FOREIGN KEY ("DOCTOR_ID")
REFERENCES "TEAM4"."USERS" ("ID") ENABLE;
```

4.2.3

a) DDL for DOCTORS_COMMENTS Table

```
CREATE TABLE "TEAM4"."DOCTORS_COMMENTS"  
(  
    "ID" NUMBER(10,0),  
    "DOCTOR_COMMENTS" VARCHAR2(255 CHAR),  
    "NURSENAME" VARCHAR2(255 CHAR),  
    "PATIENTS_NAME" VARCHAR2(255 CHAR)  
)
```

b) Constraints for DOCTORS_COMMENTS Table

```
ALTER TABLE "TEAM4"."DOCTORS_COMMENTS" MODIFY ("ID" NOT NULL ENABLE);  
ALTER TABLE "TEAM4"."DOCTORS_COMMENTS" MODIFY ("DOCTOR_COMMENTS" NOT NULL ENABLE);  
ALTER TABLE "TEAM4"."DOCTORS_COMMENTS" MODIFY ("NURSENAME" NOT NULL ENABLE);  
ALTER TABLE "TEAM4"."DOCTORS_COMMENTS" MODIFY ("PATIENTS_NAME" NOT NULL ENABLE);  
ALTER TABLE "TEAM4"."DOCTORS_COMMENTS" ADD PRIMARY KEY ("ID")
```

4.2.4

a) Constraints for HOSPITAL Table

```
ALTER TABLE "TEAM4"."HOSPITAL" MODIFY ("ID" NOT NULL ENABLE);  
ALTER TABLE "TEAM4"."HOSPITAL" MODIFY ("HOSPITALNAME" NOT NULL ENABLE);  
ALTER TABLE "TEAM4"."HOSPITAL" ADD PRIMARY KEY ("ID")
```

4.2.5

a) DDL for AVAILABILITY Table

```
CREATE TABLE "TEAM4"."AVAILABILITY"  
(  
  "ID" NUMBER(10,0),  
  "BED_AVAILABILITY" VARCHAR2(255 CHAR),  
  "DOCTORS_AVAILABILITY" VARCHAR2(255 CHAR),  
  "LOCATION" VARCHAR2(255 CHAR),  
  "HOSPITAL_ID" NUMBER(10,0),  
  "HOSPITAL_NAME" VARCHAR2(255 CHAR)  
)
```

c) Constraints for AVAILABILITY Table

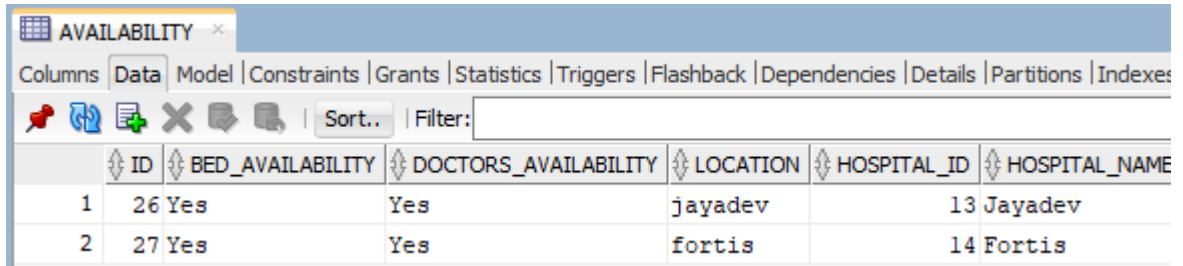
```
ALTER TABLE "TEAM4"."AVAILABILITY" MODIFY ("ID" NOT NULL ENABLE);  
ALTER TABLE "TEAM4"."AVAILABILITY" MODIFY ("HOSPITAL_ID" NOT NULL ENABLE);  
ALTER TABLE "TEAM4"."AVAILABILITY" ADD PRIMARY KEY ("ID")
```

d) Ref Constraints for AVAILABILITY Table

```
ALTER TABLE "TEAM4"."AVAILABILITY" ADD CONSTRAINT "FKD07PSKJQFKM1J0J6HB6F8WVA0" FOREIGN KEY ("HOSPITAL_ID")  
REFERENCES "TEAM4"."HOSPITAL" ("ID") ENABLE;
```

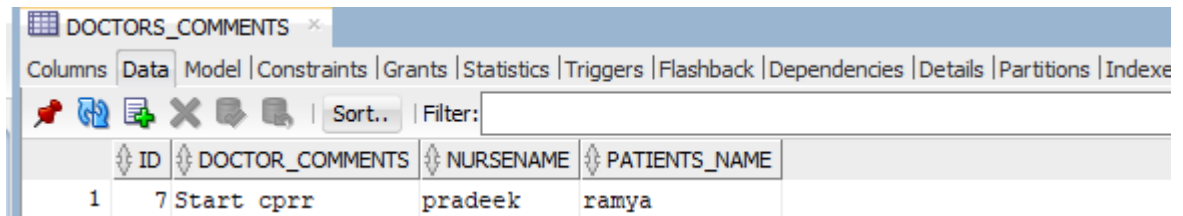
4.3 DATABASE TABLE DESIGNS

➤ AVAILABILITY TABLE



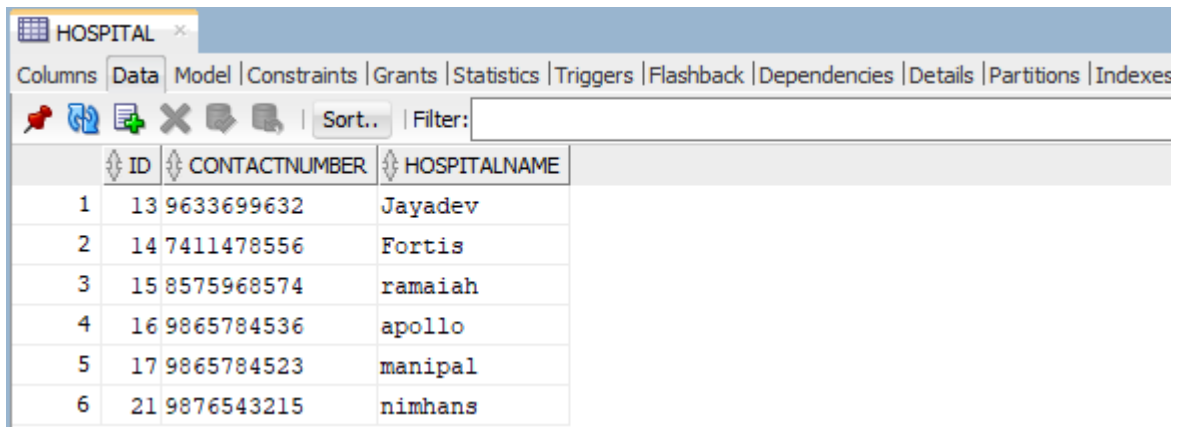
	ID	BED_AVAILABILITY	DOCTORS_AVAILABILITY	LOCATION	HOSPITAL_ID	HOSPITAL_NAME
1	26	Yes	Yes	jayadev	13	Jayadev
2	27	Yes	Yes	fortis	14	Fortis

➤ DOCTORS_COMMENTS TABLE



	ID	DOCTOR_COMMENTS	NURSENAME	PATIENTS_NAME
1	7	Start cpr	pradeek	ramya

➤ HOSPITAL TABLE



	ID	CONTACTNUMBER	HOSPITALNAME
1	13	9633699632	Jayadev
2	14	7411478556	Fortis
3	15	8575968574	ramaiah
4	16	9865784536	apollo
5	17	9865784523	manipal
6	21	9876543215	nimhans

➤ PATIENT_DETAILS TABLE

PATIENT_DETAILS										
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL										
ID	AGE	BLOOD_PRESSURE	BLOODGROUP	DISEASE	NAME	PDATE	PULSE_RATE	DOCTOR		
1	9	12 120-129/80-84mmHg	B-	Loss of blood due to accident	yashu	28-DEC-22 02.56.27.304000000	PM 68			
2	10	23 130-139/85-89mmHg	B-	low bp	divya	28-DEC-22 02.57.00.542000000	PM 68			
3	11	22 120-129/80-84mmHg	A-	Chest Pain	nayana	28-DEC-22 02.57.29.913000000	PM 76			
4	12	26 <120/<80mmHg	A+	low bp	shruthii	28-DEC-22 02.58.24.097000000	PM 77			
5	28	21 130-139/85-89mmHg	B-	fever	akshay	29-DEC-22 09.55.19.357000000	AM 96			
6	45	21 <120/<80mmHg	A+	severe stomach pain	Doctor	04-JAN-23 03.40.28.212753000	PM 75			
7	46	21 <120/<80mmHg	A-	severe stomach pain	rksacqd	04-JAN-23 03.41.19.822490000	PM 75			
8	49	28 <120/<80mmHg	A-	severe stomach pain	shruru	04-JAN-23 05.08.03.667889000	PM 79			
9	51	22 120-129/80-84mmHg	B+	fever	pradeekk	04-JAN-23 05.21.47.666000000	PM 74			

➤ USERS TABLE

USERS									
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL									
ID	EMAIL	NAME	OTP	PASSWORD	PNUMBER	ROLE			
1	1 shreyasonu25@gmail.com	shreya	647950	0 \$2a\$10\$TeTr837sdEOyy5cUviaDe./De1ZvmS5gU4TRS8kA6oJjCDdWyp8bi	9898768976	DOCTOR	s		
2	2 syedaheenal433@gmail.com	heena	478394	0 \$2a\$10\$r15Y9UV3CbOgNvvkNh0V..Cit74.LEFT9NB5Cc7F7hsVO9NWujnwy	7412035698	DOCTOR	h		
3	3 its.shwethac@gmail.com	shwetha	0	0 \$2a\$10\$leloP2xd5mQR6bLNU8FaZuRjJ.WM2BThZPpriaOoxsThiIRJTGydm	7896541200	DOCTOR	s		
4	4 iampradeek@gmail.com	pradeek	0	0 \$2a\$10\$Wj5/2WPPBXqTk3M/w2DxP.FMYN8tXzBL8M3lyCHtT1AedTKVU1B2G	7411478555	NURSE	p		
5	5 itsajayrawal2000@gmail.com	ajayy	0	0 \$2a\$10\$Qt7RtysTvPnqWdzRvdY73ut4u4NBcIqAJdb07HNizlsJ3sVX9y/32	8745541000	NURSE	a		
6	6 csaisubramanyaml@gmail.com	subbu	0	0 \$2a\$10\$isRUj1xdZaTjyyQXordp6OHY/f4RwDDNA2BuLZafkNWVPtSHluRSy	9633699633	NURSE	s		
7	42 aaaaa@gmail.com	aaaaa	0	0 \$2a\$10\$3YXtel9wuBplYxrBmU49iuMBnJcmYfJYU8NwZ5BzR4ZMcpR1K.EG.	9876543210	DOCTOR	a		
8	43 doctor@gmail.com	Doctor	0	0 \$2a\$10\$zgVL3v17DUxlGaknYIB7eOhoNVpYF2191eu0Edg.h4Jt8pwVF6w4m	9876543210	DOCTOR	d		
9	47 ronaldo@gmail.com	ronaldo	0	0 \$2a\$10\$pfDvy.DhEVQLyVgk2cy.peNN9LevRG8cA4J7kuEB22n.Y19uUHWau	9874560123	DOCTOR	r		
10	48 messi@gmail.com	messi	0	0 \$2a\$10\$HGfeteVegRO1vIS3JtKJ.34yKw3Z8TsnK/k1Y2VVe2nai0jaiymq	9517864236	NURSE	m		
11	29 shruthid017@gmail.com	shruthi	0	0 \$2a\$10\$mwuo5ztMrCOIZ3Fe4TSYZuKI.B6kfx2bAAYm9dS9VKT2sIDBhsfRS	9685745869	DOCTOR	s		
12	44 sanju@gmail.com	sanju	0	0 \$2a\$10\$bG5WludF9oxTuDtctL/vUuKN7N5ARoY2AZipwAWlafJM0RpK3z1Gm	7891234560	NURSE	s		

CHAPTER 5

SNAPSHOTS

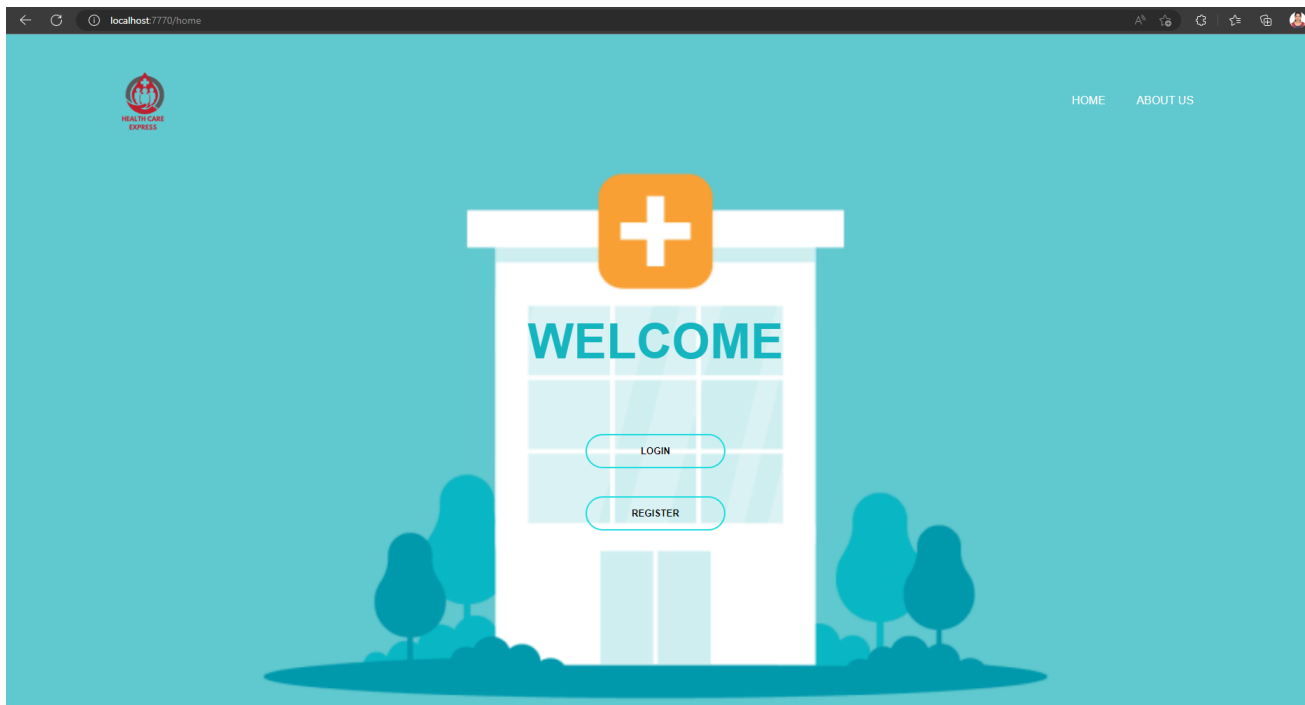


Fig 5.1 HOME PAGE

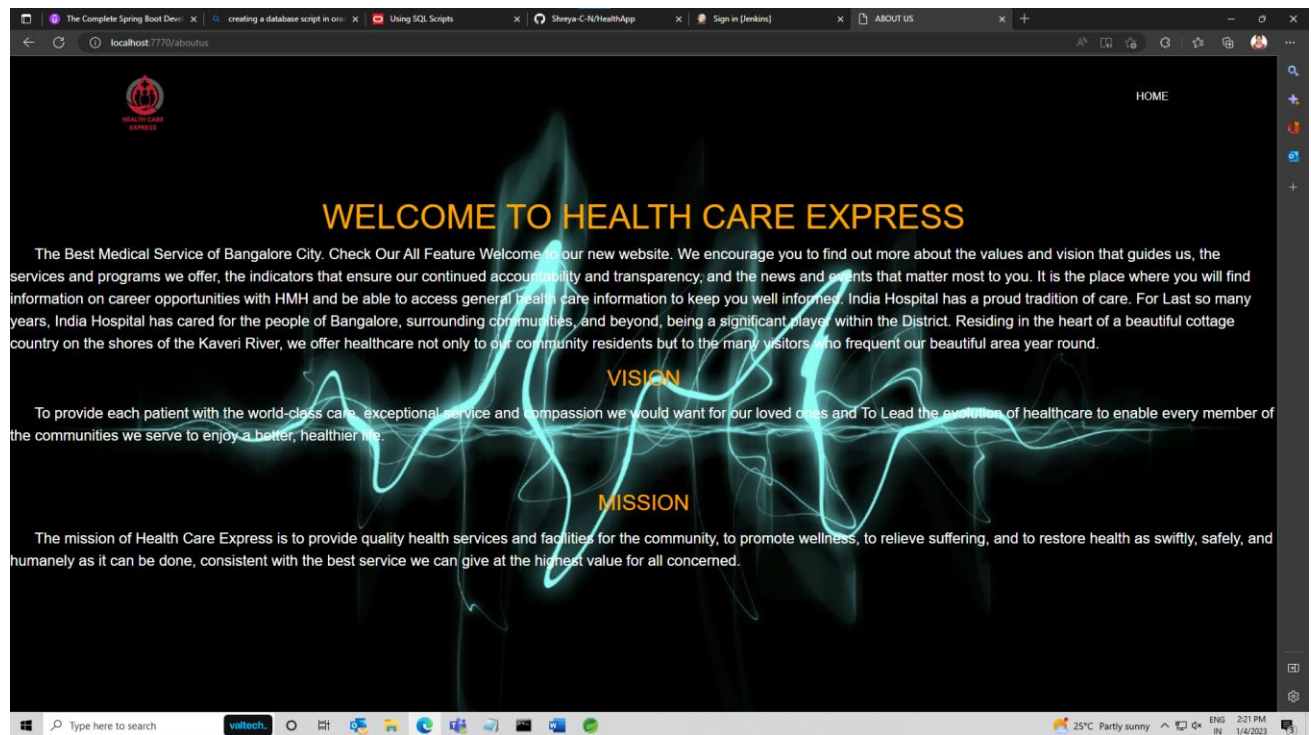


Fig 5.2 ABOUTUS PAGE

The screenshot shows a web browser window with the URL `localhost:7770/register/submit=REGISTER`. The page features a header with a logo on the left and links for [HOME](#) and [ABOUT US](#) on the right. The main content area displays a registration form titled "REGISTRATION FORM" in a blue box. The form includes input fields for Name, Phone Number, Email, Username, Password, and Confirm Password. Below these fields is a dropdown menu labeled "Select Your Role". At the bottom of the form are two blue buttons: "REGISTER" and "CLEAR", and a link for [ALREADY REGISTERED USER](#). The background of the page is an illustration of a healthcare setting with a receptionist at a desk and a patient sitting on a couch.

FIG 5.3 REGISTRATION PAGE

This screenshot shows the same registration page as Figure 5.3, but with the form fields populated with a doctor's information. The "Name" field contains "Doctor", "Phone Number" contains "9876543210", "Email" contains "doctor@gmail.com", "Username" contains "doctor", "Password" and "Confirm Password" both contain "*****". The "Select Your Role" dropdown menu is set to "DOCTOR". The "REGISTER" and "CLEAR" buttons remain visible at the bottom of the form, along with the [ALREADY REGISTERED USER](#) link. The browser window and page layout are identical to the previous figure.

FIG 5.4.1 DOCTOR REGISTRATION

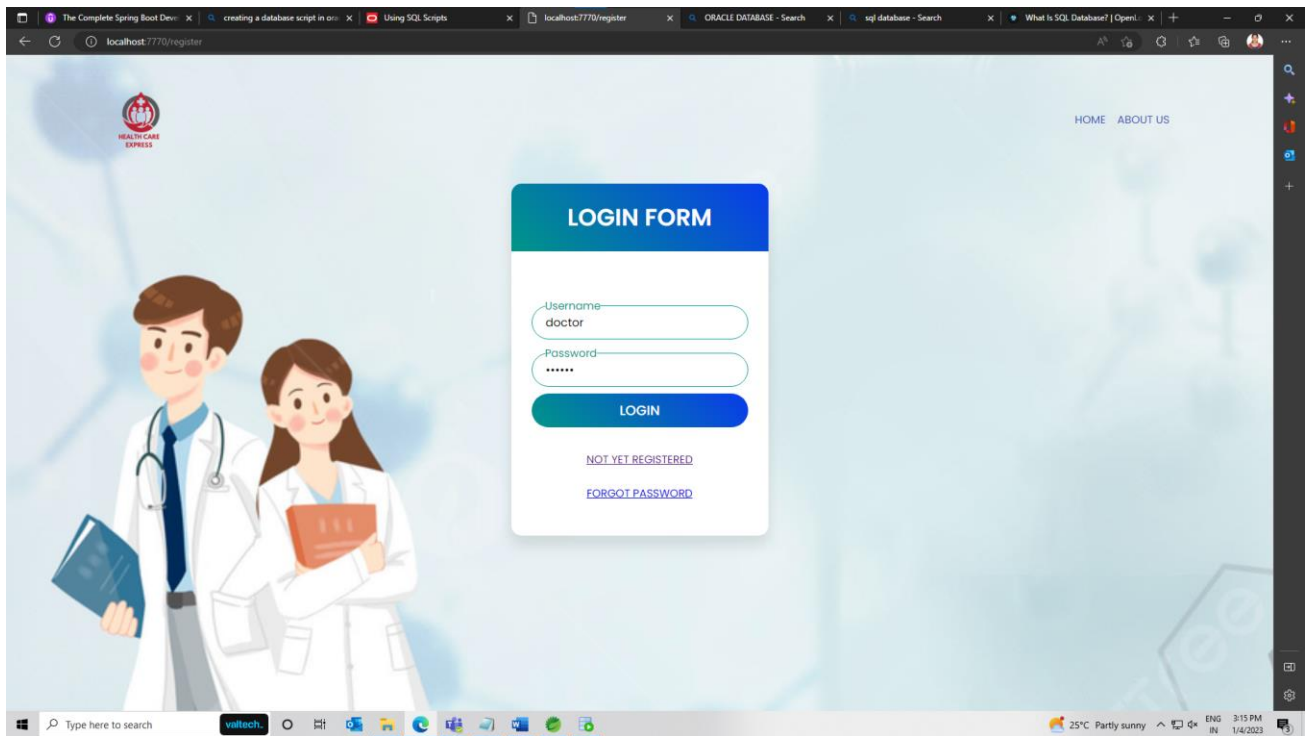


FIG 5.4.2 DOCTOR LOGIN

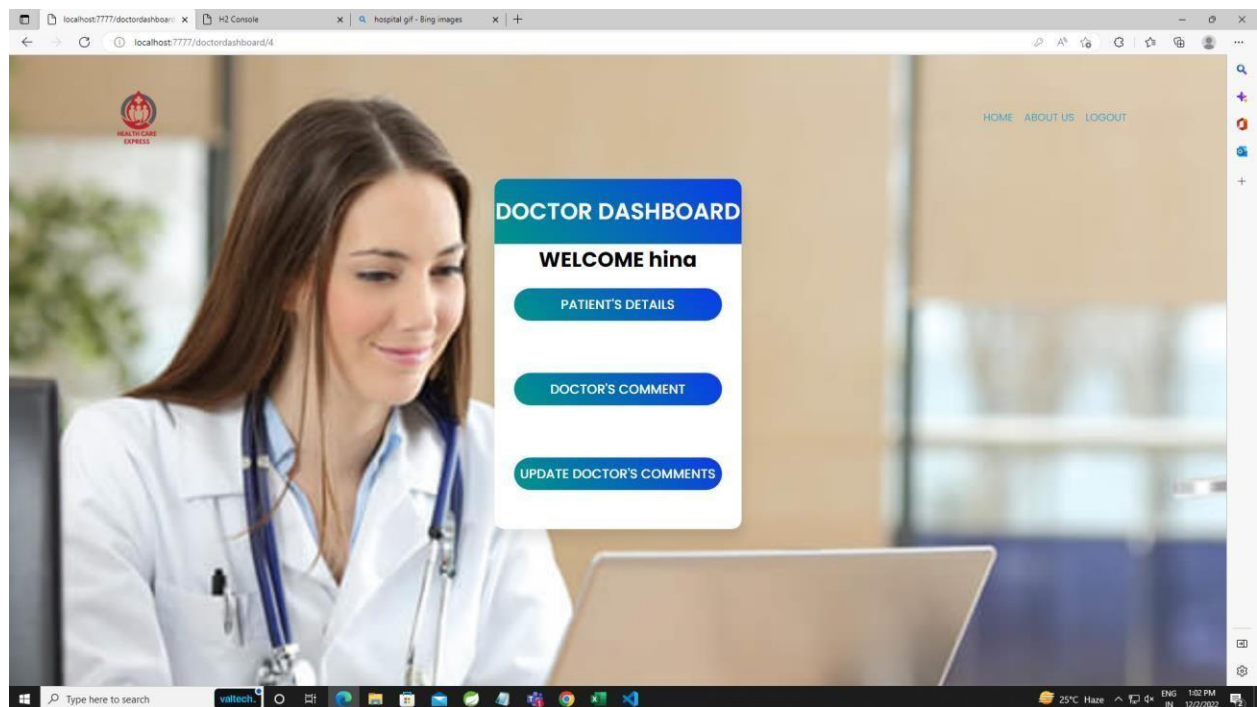
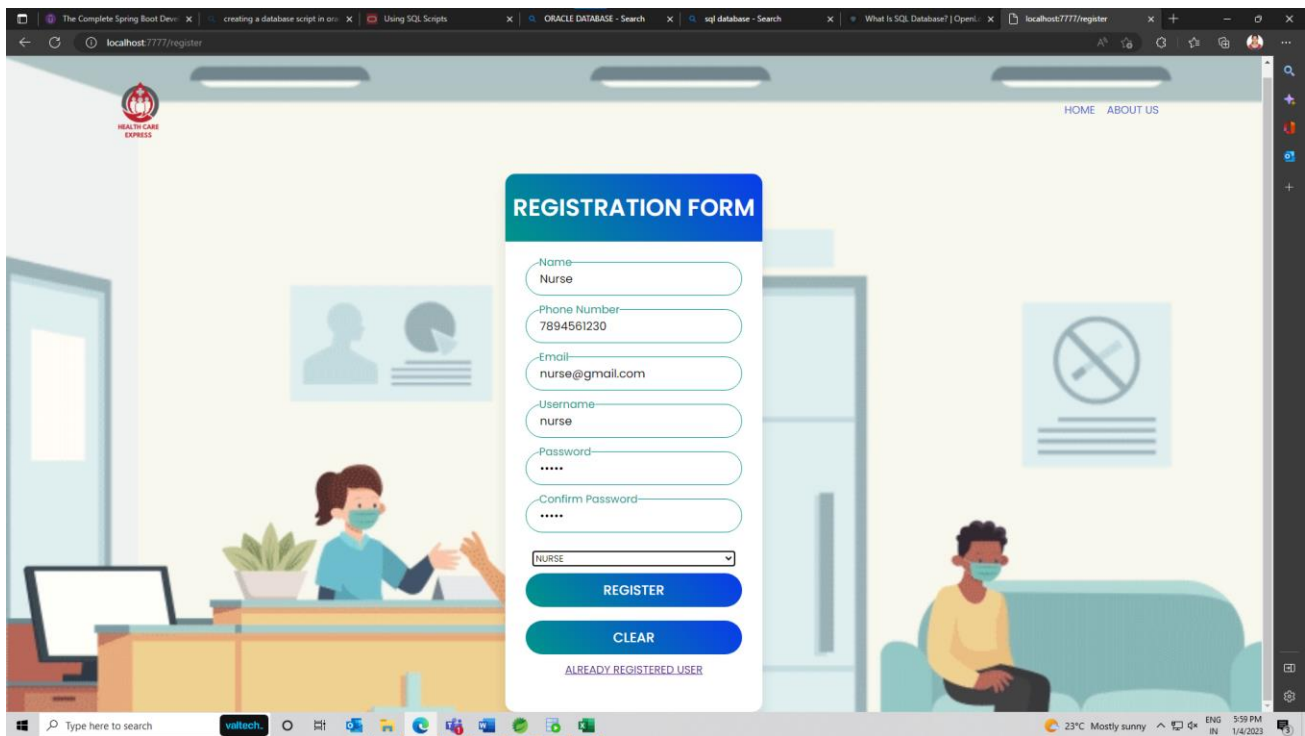
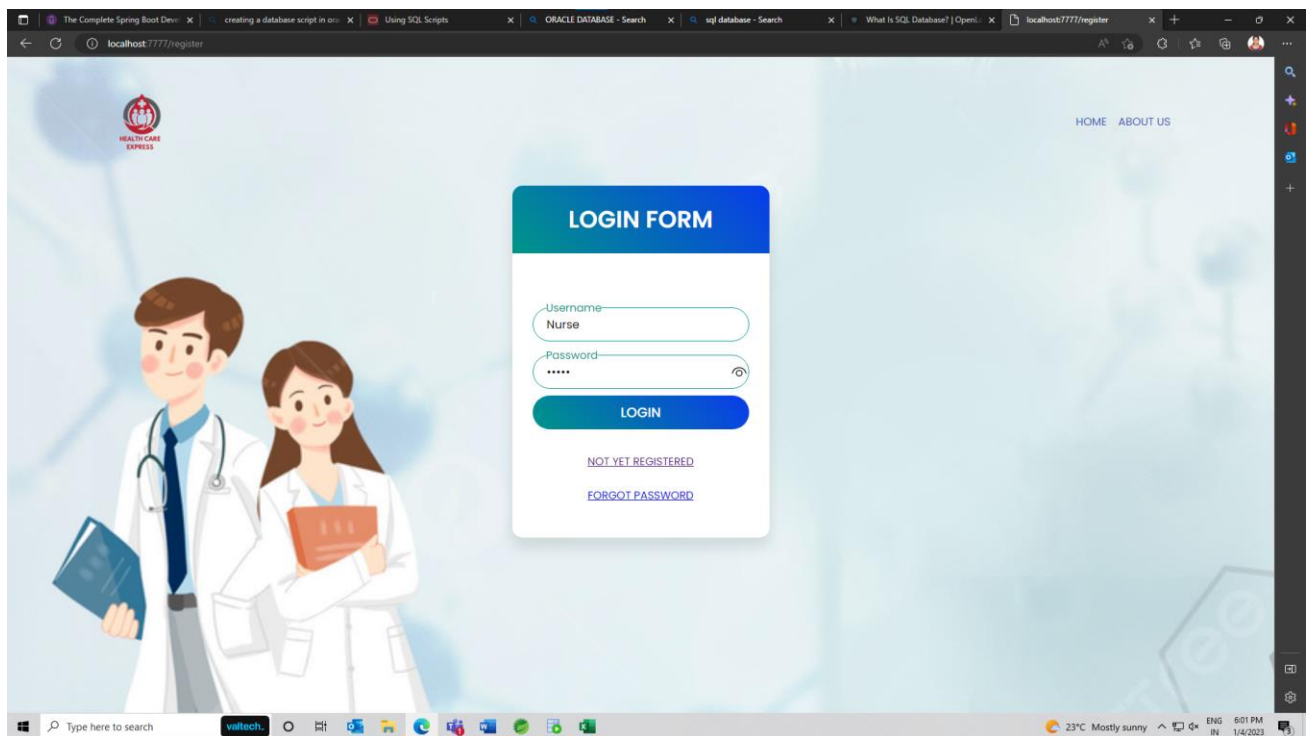


FIG 5.4.3 DOCTOR DASHBOARD PAGE



The screenshot shows a web browser window with the URL `localhost:7777/register`. The page features a header with a logo and navigation links for `HOME` and `ABOUT US`. The main content area is titled `REGISTRATION FORM` and contains the following fields: `Name` (with a dropdown menu set to `Nurse`), `Phone Number` (with the value `7894561230`), `Email` (with the value `nurse@gmail.com`), `Username` (with the value `nurse`), `Password` (masked with dots), and `Confirm Password` (masked with dots). Below these fields are `REGISTER` and `CLEAR` buttons, and a link for `ALREADY REGISTERED USER`. The background illustration depicts a healthcare setting with a nurse at a desk and a patient sitting on a couch.

FIG 5.5 NURSE REGISTRATION PAGE



The screenshot shows a web browser window with the URL `localhost:7777/register`. The page features a header with a logo and navigation links for `HOME` and `ABOUT US`. The main content area is titled `LOGIN FORM` and contains the following fields: `Username` (with the value `Nurse`) and `Password` (masked with dots). Below these fields is a `LOGIN` button. There are also links for `NOT YET REGISTERED` and `FORGOT PASSWORD`. The background illustration depicts two healthcare professionals, a male doctor and a female nurse, holding medical equipment.

FIG 5.5 NURSE LOGIN PAGE

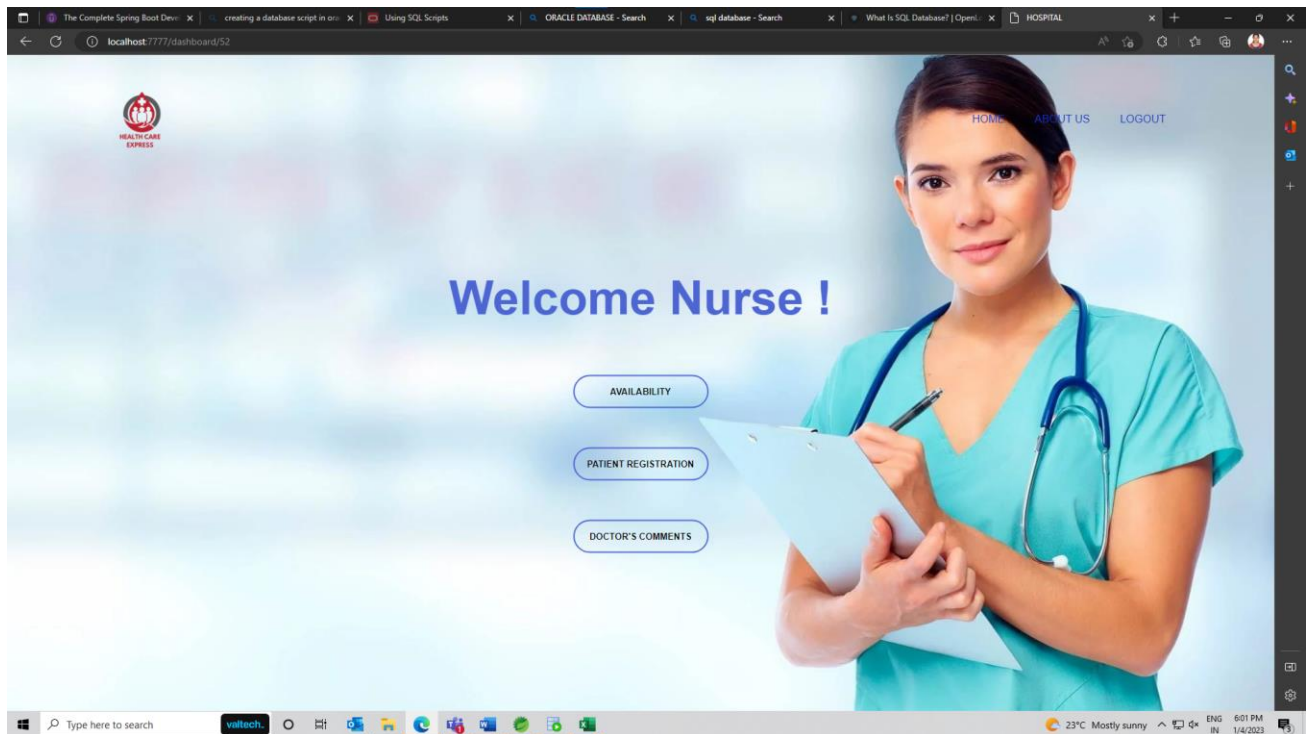


FIG 5.6 NURSE DASHBOARD

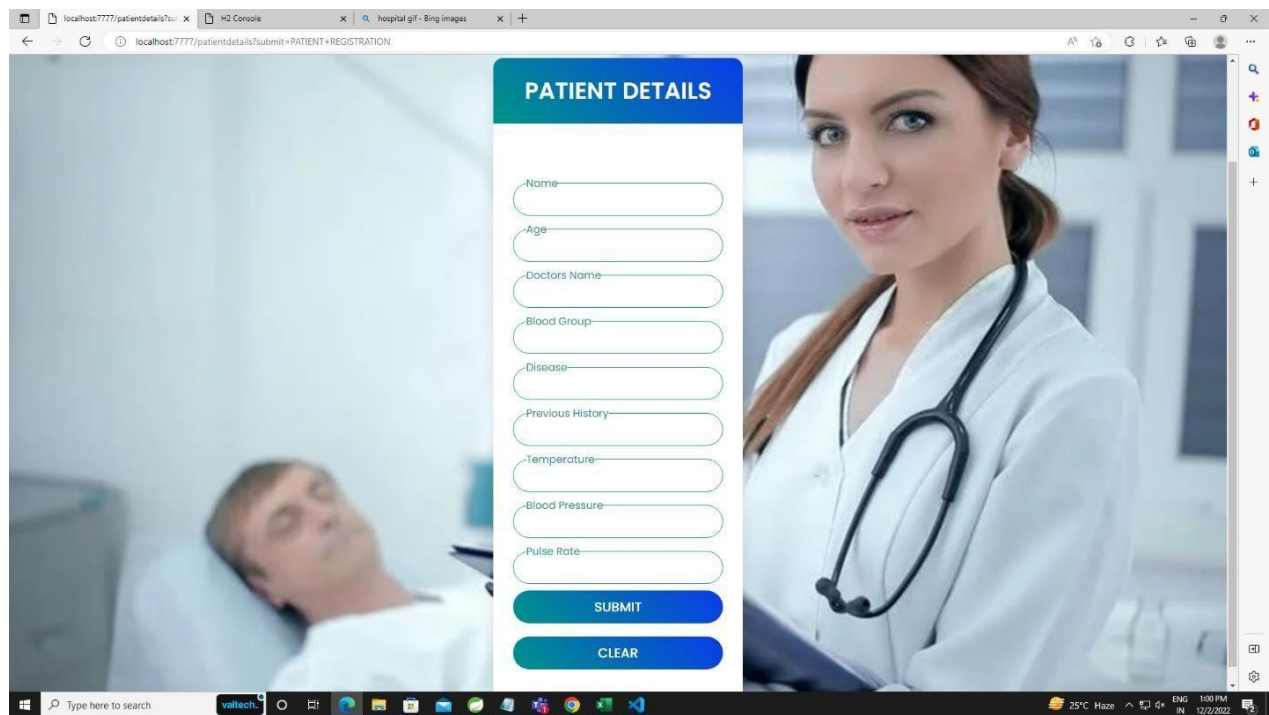


FIG 6.1 PATIENT DETAILS PAGE

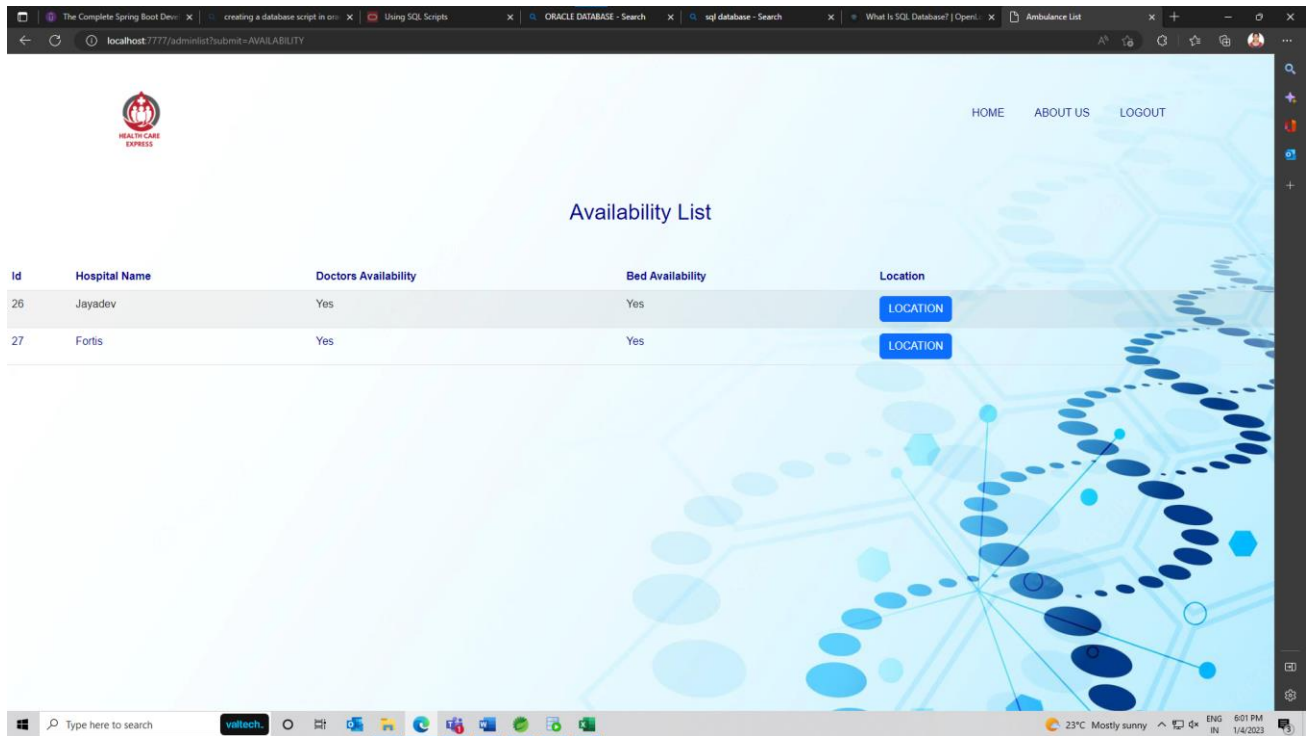


FIG 6.2 LIST OF PATIENTS PAGE

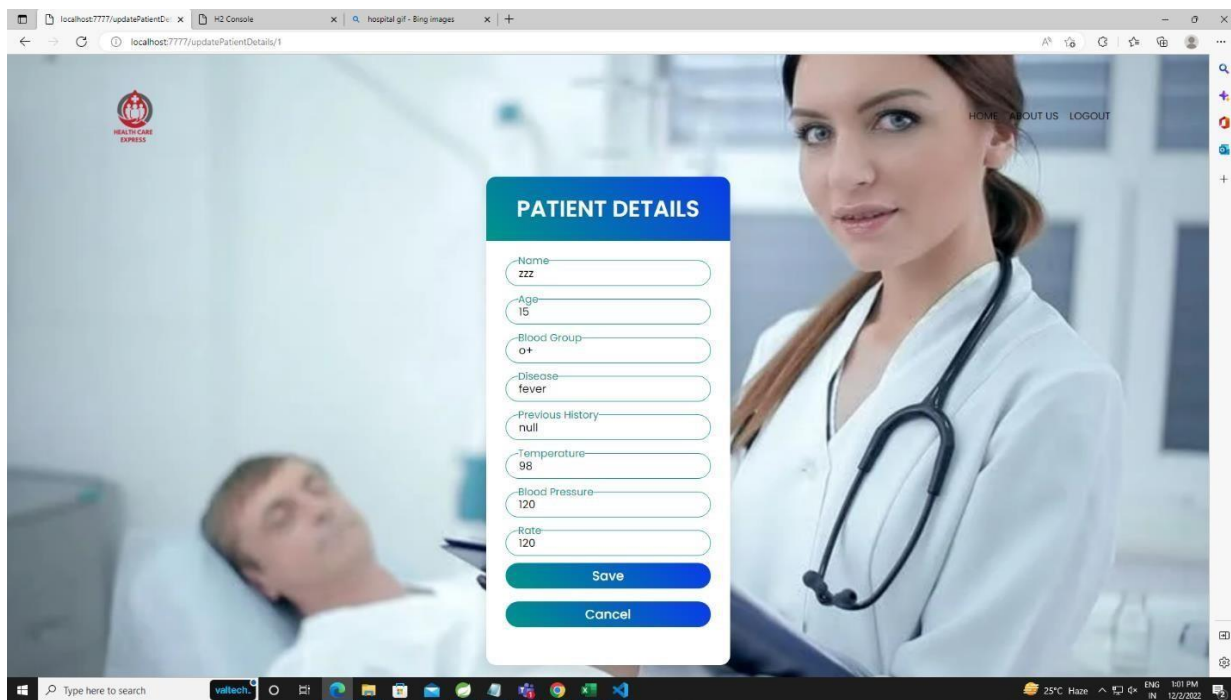


FIG 6.3 UPDATE PATIENT DETAILS PAGE

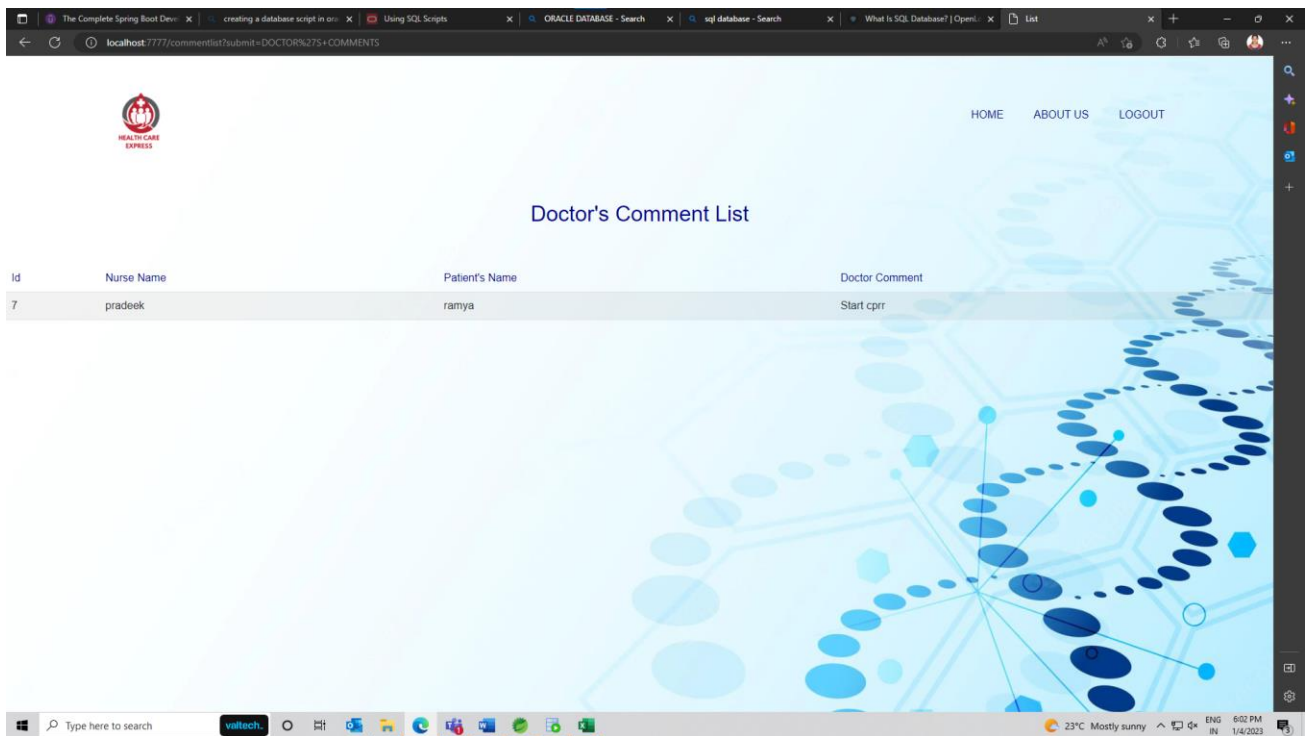


FIG 6.4 DOCTOR'S COMMENT ON PATIENT PAGE

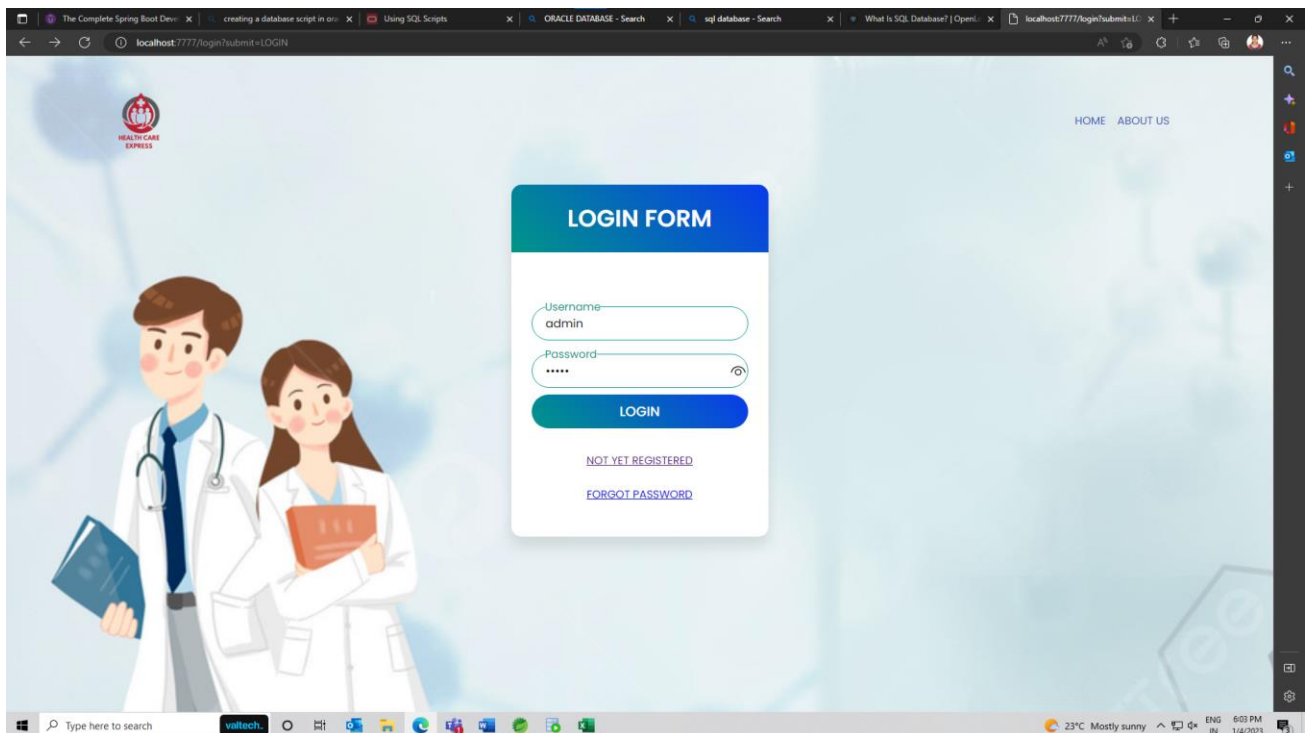


FIG 7 ADMIN LOGIN PAGE

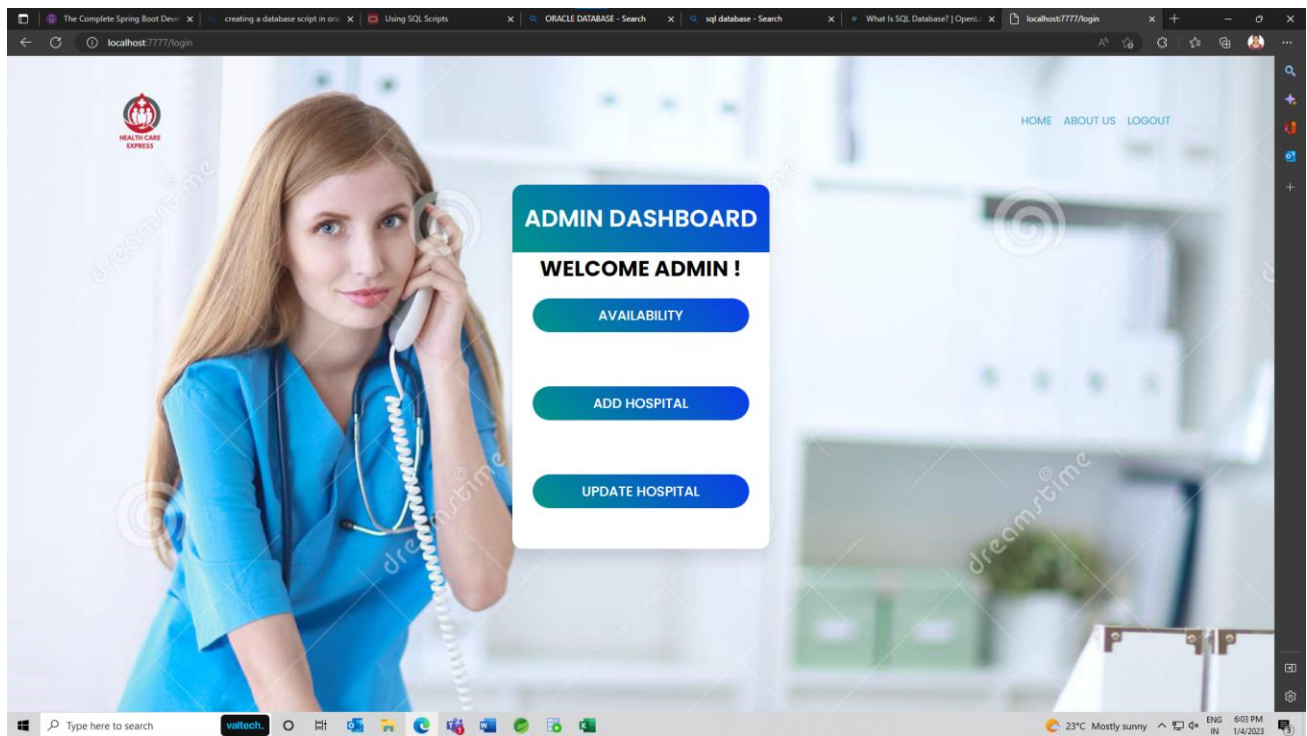


FIG 7.1 ADMIN DASHBOARD PAGE

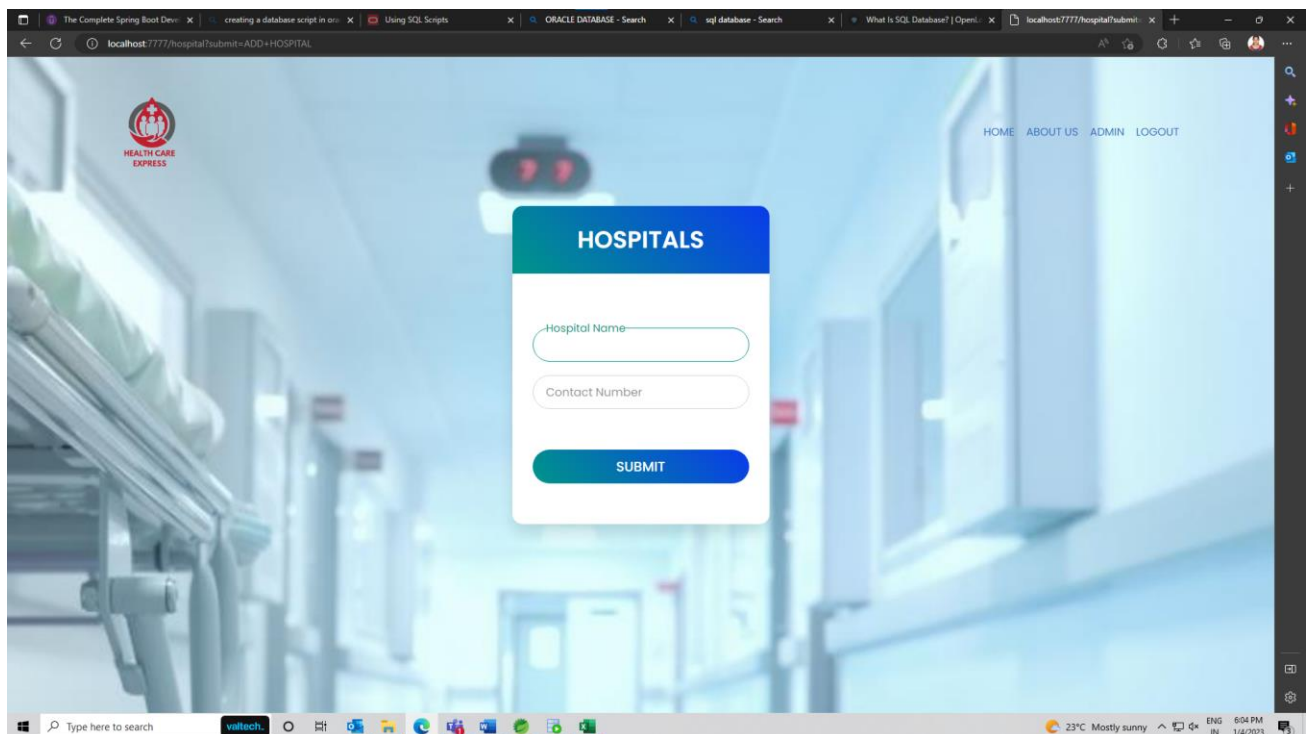


FIG 7.2 ADMIN UPDATE PAGE