

**A
REPORT
On
WEB DEVELOPMENT**

Submitted

In partial fulfillment

For the award of the degree of

Submitted By:

G.Padmaja naidu(221710306021)

G.Akshaya reddy(221710314017)

B.Shashank(221710314004)

Submitted To:

Dhyanahita,

Full stack development team.

Department of Computer Science and Engineering

CANDIDATES DECLARATION

We here by declare that the project entitled "MEDICINO(Hospital management) submitted by us to Dhyanahitha institution in partial fulfillment of the requirements for the certificate of internship on fullstack development is a record of bonafide project work carried out by us under guidance of our mentor.we further declare that the work reported in this project has not been submitted and will not be submitted,either in part or in full,for the award or certificate of any other degree .

Signature of candidates

G.Akshaya Reddy(221710314017)

B.Shashank(221710314004)

G.Padmaja naidu(221710306021)

ACKNOWLEDGEMENT

We would like to express our special profound gratitude to my trainer as well as our mentor who gave us the golden opportunity to do this wonderful project on the topic (Hospital management), which also helped us in doing a lot of Research on the bridge of practical and theoretical and we came to know about so many new things regarding different technologies we are sincerely thankful to them.

ABSTRACT

Web development is the work involved in developing a website for the Internet (World Wide Web) or an intranet (a private network). Web development can range from developing a simple single static page of plain text to complex web-based internet applications (web apps), electronic businesses, and social network services. A more comprehensive list of tasks to which web development commonly refers, may include web engineering, web design, web content development, client liaison, client-side/server-side scripting, web server and network security configuration, and e-commerce development.

Among web professionals, "web development" usually refers to the main non-design aspects of building websites: writing markup and coding. Web development may use content management systems (CMS) to make content changes easier and available with basic technical skills.

For larger organizations and businesses, web development teams can consist of hundreds of people (web developers) and follow standard methods like Agile methodologies while developing websites. Smaller organizations may only require a single permanent or contracting developer, or secondary assignment to related job positions such as a graphic designer or information systems technician. Web development may be a collaborative effort between departments rather than the domain of a designated department. There are three kinds of web developer specialization: front-end developer, back-end developer, and full-stack developer. Front-end developers are responsible for behavior and visuals that run in the user browser, while back-end developers deal with the servers.

Table of content

	Title	Page No.
1.	Introduction..... 1.1 objectives	9
2.	Web development..... 2.1 website 2.2 webpage	11-17
3.	The steps to create a website..... 3.1 UI development 3.1.1 HTML 3.1.2 CSS 3.1.3 Bootstrap 3.2 Scripting 3.3 Database 3.3.1 SQL 3.3.2 Query	17-33
4	Scripting languages..... 4.1 Java script 4.2 J Query	33-34
5	Requirements..... 5.1 software requirements 5.2 hardware requirements	34-35
6	About the project.....	35-36
7	Modules..... 7.1 registration page 7.2 login page	36-37
8	Workflow.....	37-38
9	Outputs and source codes of the project.....	39-111

10.	Maintenance.....	111-112
11.	Future scope and enhancement.....	112
12.	Conclusion.....	112

List of Figures

Figure.1.1	full stack development.....
Figure.2.1	steps for web design.....
Figure.2.2	pages in website.....
Figure.2.3	creation of a web page.....
Figure.3.1	website development.....
Figure.3.2	Types of CSS.....
Figure.3.3	login page before CSS.....
Figure.3.4	login page after CSS.....
Figure.3.5	installation of bootstrap.....
Figure.3.6	selection of bootstrap version.....
Figure.3.7	output of html file after linking with bootstrap..
Figure.3.8	popularity of programming languages in2020..
Figure.3.9	client side scripting.....
Figure.3.10	database.....
Figure.3.11	SQL server.....
Figure.8.1	workflow diagram of hospital management.....

Figure.9.1 to 9.14	Outputs of the project.....
--------------------	-----------------------------

List of tables

Table.2.1	phases of web development.....
Table.3.1	html tags.....
Table.5.1	software requirements.....
Table.5.2	hardware requirements.....

CHAPTER-1

INTRODUCTION



fig.1.1

A full stack developer is a web developer or engineer who works with both the front and back ends of a website or application—meaning they can tackle projects that involve databases, building user-facing websites, or even work with clients during the planning phase of projects.

Full stack web developers:

- Are familiar with HTML, CSS, JavaScript, for frontend developing and for backend languages like SQL for database, JQUERY, bootstrapping etc

Full stack development is done between frontend versus backend which can create more visualizations and database access.

So full stack development is classified into two phases :

1. Frontend making
2. Backend making

FRONTEND:

Front end developers build the visible parts of websites that users see and interact within their web browsers.

The front end of a website (or web or mobile application) is the part a user sees and directly interacts with. The front end is built with languages like:

- HTML (HyperText Markup Language)
- CSS (cascading style sheet)
- JavaScript
- JQuery
- Bootstrap

BACKEND:

Back end developers build the “under the hood” parts of websites that users don’t interact with directly.

So what does this mean for front end vs back end? While the front end is everything the user interacts with directly, the back end is much more behind-the-scenes and can have some advantages over front end technologies for specific projects. Back end programming languages include:

- SQL
- Queries
- Database
- Php

Back end developers generally work with a front end developer to make their code work within the site or app design (or to tweak that design when necessary) and front end. Which finally brings us to the topic of full stack

.

1.1 OBJECTIVES

Designed and developed efficient and maintainable software according to the business **objectives** and needs of the company's various clients. Designed and developed effective app-based solutions to address the problems and concerns of the company's clients. Complete development with client requirements and reasonable budget .it should satisfy all requirements of clients.

CHAPTER -2

WEB DEVELOPMENT

Web development is the building and maintenance of websites; it's the work that happens behind the scenes to make a website look great, work fast and perform well with a seamless user experience.

Web developers, or 'devs', do this by using a variety of coding languages. The languages they use depends on the types of tasks they are performing and the platforms on which they are working.

Web development skills are in high demand worldwide and well paid too – making development a great career option. It is one of the easiest accessible higher paid fields as you do not need a traditional university degree to become qualified. Every Web Developer must have a basic understanding of HTML, CSS, and JavaScript. Responsive Web Design is used in all types of modern web development

2.1 Life cycle of web development

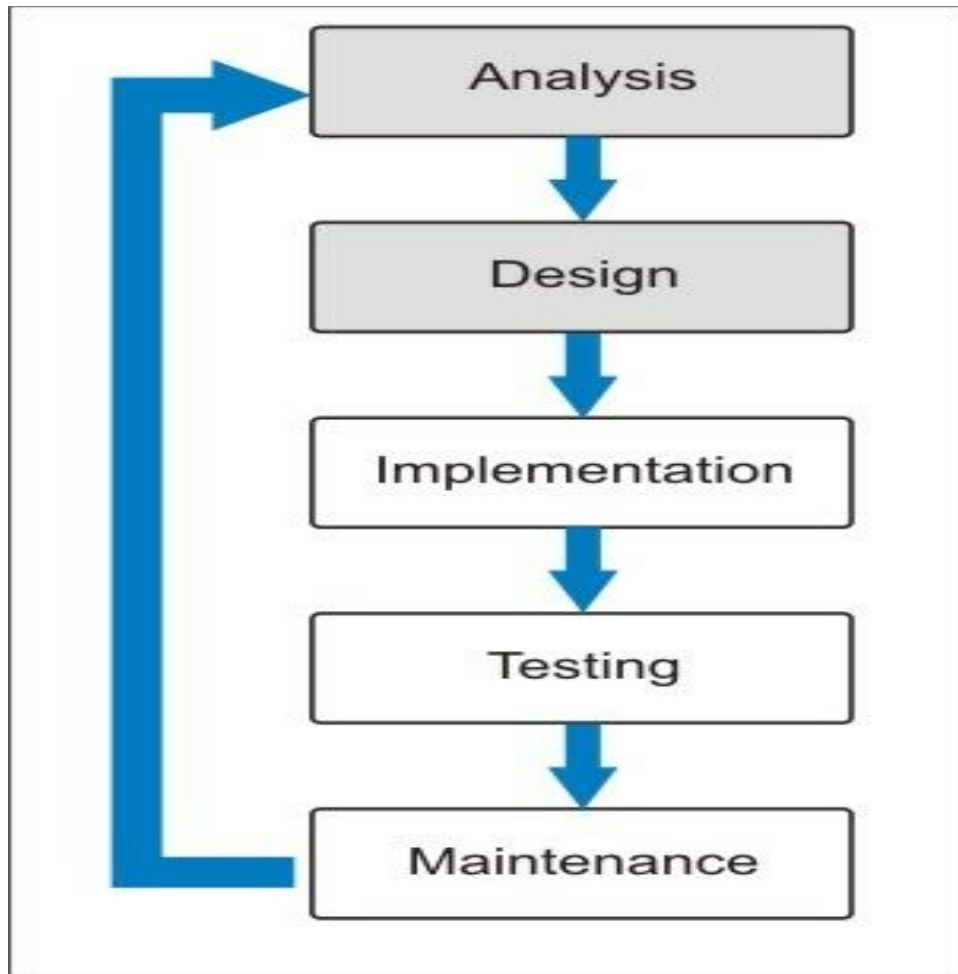


fig.2.1

Web Development Phases and Questions	
Web development phase	Questions to ask
Planning	<ul style="list-style-type: none"> • What is the purpose of the Web site • Who will use this Web site • What are their computing environment • Who owns and authors the information on the Web site

Analysis	<ul style="list-style-type: none"> • What information is useful to the user
Design and development	<ul style="list-style-type: none"> • What type of Website layout is appropriate • What forms of multimedia is helpful to the user
Testing	<ul style="list-style-type: none"> • Is the Web site content correct • Does the Website functions correctly • Are users able to find the information they need • Is the navigation easy to use?
Implementation and Maintenance	<ul style="list-style-type: none"> • How is the Website published • How is the Website updated • Who is responsible for content updates • Will the Web site be monitored

Table.2.1

2.2 WEB-SITE

What exactly is a website and who needs one?websites are without doubt the most important element of the internet.

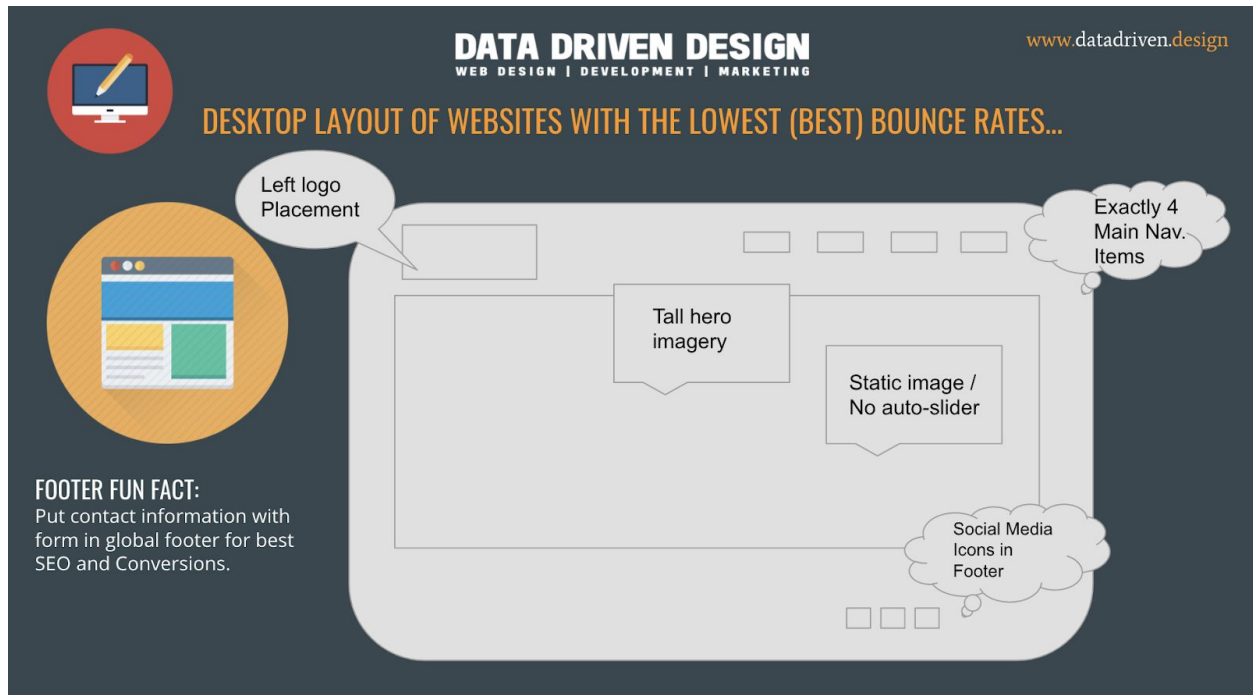


fig.2.2

A **website** is a collection of web pages and related content that is identified by a common domain name and published on at least one web server. Notable examples are wikipedia.org, google.com, and amazon.com.

All publicly accessible websites collectively constitute the World Wide Web. There are also private websites that can only be accessed on a private network, such as a company's internal website for its employees.

Websites are typically dedicated to a particular topic or purpose, such as news, education, commerce, entertainment, or social networking. Hyperlinking between web pages guides the navigation of the site, which often starts with a home page.

Users can access websites on a range of devices, including desktops, laptops, tablets, and smartphones. The software application used on these devices is called a web browser.

The central page of a website is called a home page. This is usually the first page you see when you call a website up and can also be called a 'start page' or 'index page'. From here onwards, the user delves into the site's subpages like login page,registration page,feedback page,etc according to website concept.

2.2 WEB-PAGE

A web page (or webpage) is a specific collection of information provided by a website and displayed to a user in a web browser. A website typically consists of many web pages linked together in a coherent fashion. The name "web page" is a metaphor of paper pages bound together into a book.

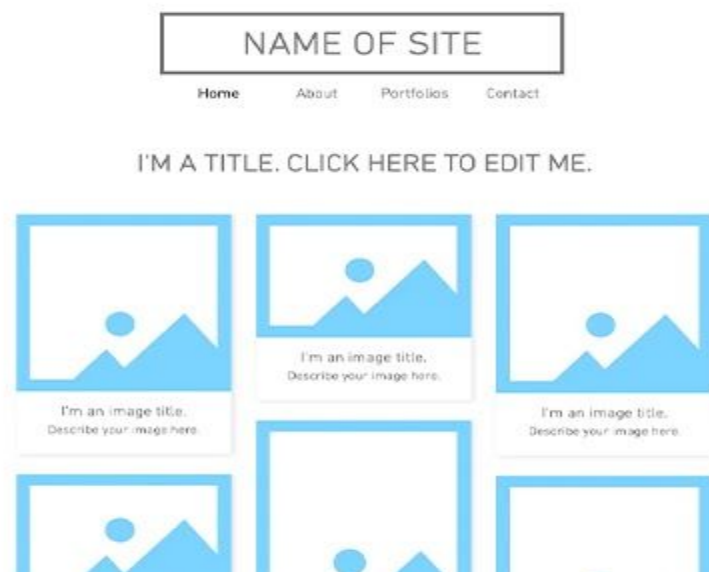


fig.2.3

Web page consists of:

- 1.Elements
- 2.Navigation
- 3.Deployment

Elements:

The core element of a web page is one or more text files written in the Hypertext Markup Language (HTML). Many web pages also make use of JavaScript code for dynamic behavior and Cascading Style Sheets (CSS) code for presentation semantics. Images, videos, and other multimedia files are also often embedded in web pages.

Navigation:

Each web page is identified by a distinct Uniform Resource Locator (URL). When the user inputs a URL into their browser, that page's elements are downloaded from web servers. The browser then transforms all of the elements into an interactive visual representation on the user's device.

If the user clicks or taps a link to another page, the browser repeats this process to display the new page, which could be part of the current website or a different one.

Deployment:

From the perspective of server-side website deployment, there are two types of web pages, static and dynamic. Static pages are retrieved from the web server's file system without any modification, while dynamic pages must be created by the web server on the fly, typically drawing from a database to fill out a web template, before being sent to the user's browser.

CHAPTER-3

STEPS TO CREATE A WEBSITE

Creating a web site requires multiple steps:

- 1 . Creating a UI (User interface)
2. Scripting(Both at server and client end)

3. Creating a backend or the database(includes SQL,Queries)

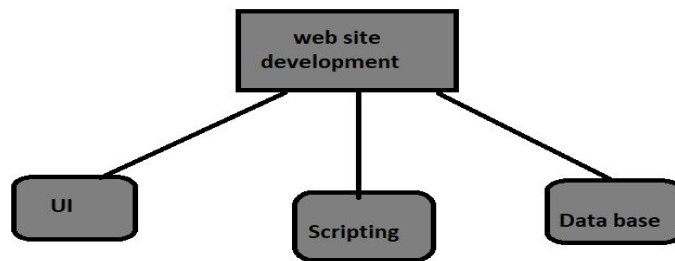


Fig.3.1

3.1 UI DEVELOPMENT

Technologies that are mostly used to develop a User Interface are:

- 1.HTML
- 2.CSS
- 3.BootStrap

3.1.1 HTML

HTML stands for **Hyper Text Markup Language**. It is used to design web pages using markup language. HTML is the combination of Hypertext and Markup language.

Hypertext defines the link between the web pages. Markup language is used to define the text document within a 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 (e.g. HTML) languages are

human readable. Language uses tags to define what manipulation has to be done on the text.

HTML is a markup language which is used by the browser to manipulate text, images and other content to display it in 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 which was published in 1999.

An HTML element is defined by a starting tag. If the element contains other content, it ends with a closing tag, where the element name is preceded by a forward slash as shown below with few tags –

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h <i>n</i> > ... </h <i>n</i> >	Delimits a level <i>n</i> heading
 ... 	Set ... in boldface
<i> ... </i>	Set ... in italics
<center> ... </center>	Center ... on the page horizontally
 ... 	Brackets an unordered (bulleted) list
 ... 	Brackets a numbered list
 ... 	Brackets an item in an ordered or numbered list
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a horizontal rule
	Displays an image here
 ... 	Defines a hyperlink

Table.3.1

So here <html>....</html>,<head>....</head>,<p>....</p> is an HTML element, <hn>...</hn> is another HTML element. There are some HTML elements which don't

need to be closed, such as <img.../>, <hr /> and
 elements. These are known as void elements. still we have many tags in the html.

HTML documents consist of a tree of these elements and they specify how HTML documents should be built, and what kind of content should be placed in what part of an HTML document.

HTML can embed programs written in scripting languages such as JavaScript which affect the behaviour and content of webpages. Inclusion of CSS define the look and layout of content.

The following is an example of the classic hello world program, a common test for comparing programming languages, scripting languages and markup languages.

Example of html source code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>page title</title>
  </head>
  <body>
    <h1>My first heading</h1>
    <p>Hello world</p>
  </body>
</html>
```

- The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the HTML page
- The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)

- The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

Output will be like:

My First Heading

Hello world

3.1.2 CSS

Cascading Style Sheets, fondly referred to as **CSS**, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

CSS is easy to learn and understood but it provides powerful control over the presentation of an HTML document.

CSS saves time you can write CSS once and reuse the same sheet in multiple HTML pages. Easy Maintenance to make a global change simply change the style, and all elements in all the webpages will be updated automatically. Search Engines of CSS is considered as a clean coding technique, which means search engines won't have to struggle to "read" its content. CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes. Offline Browsing of CSS can store web applications locally with the help of offline cache. Using of this we can view offline websites.

3.1.2 CSS

Types of CSS:

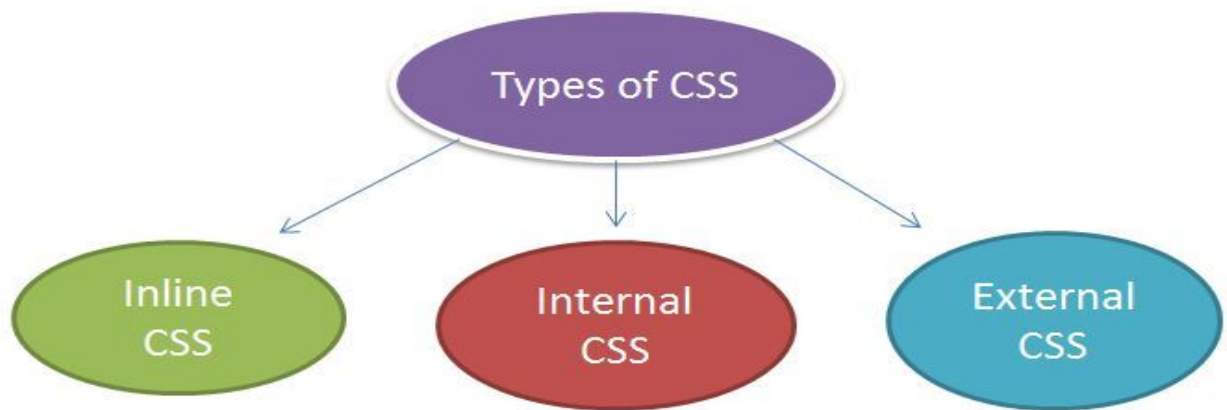


fig.3.2

Inline CSS:

InlineCSS is used to style a specific HTML element. For this CSS style, you'll only need to add the **style** attribute to each HTML tag, without using selectors. This CSS type is not really recommended, as each HTML tag needs to be styled individually. Managing your website may become too hard if you only use inline CSS. However, inline CSS in HTML can be useful in some situations. For example, in cases where you don't have access to CSS files or need to apply styles for a single element only.

Internal CSS:

Internal or embedded CSS requires you to add the **<style>** tag in the **<head>** section of your HTML document. This CSS style is an effective method of styling a single page. However, using this style for multiple pages is time-consuming as you need to put CSS rules to every page of your website.

External CSS:

With external CSS, you'll link your web pages to an external **.css** file, which can be created by any text editor in your device (e.g., **Notepad++**). This CSS type is a more efficient method, especially for styling a large website. By editing one **.css** file, you can change your entire site at once.

Login page before using CSS:

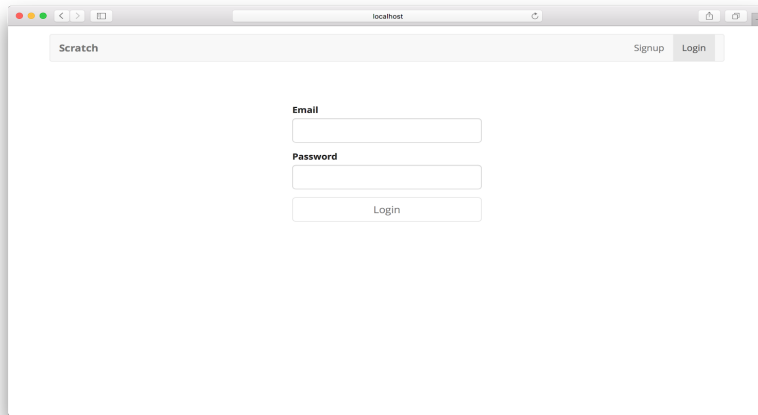


Fig.3.3

Login page after using CSS:

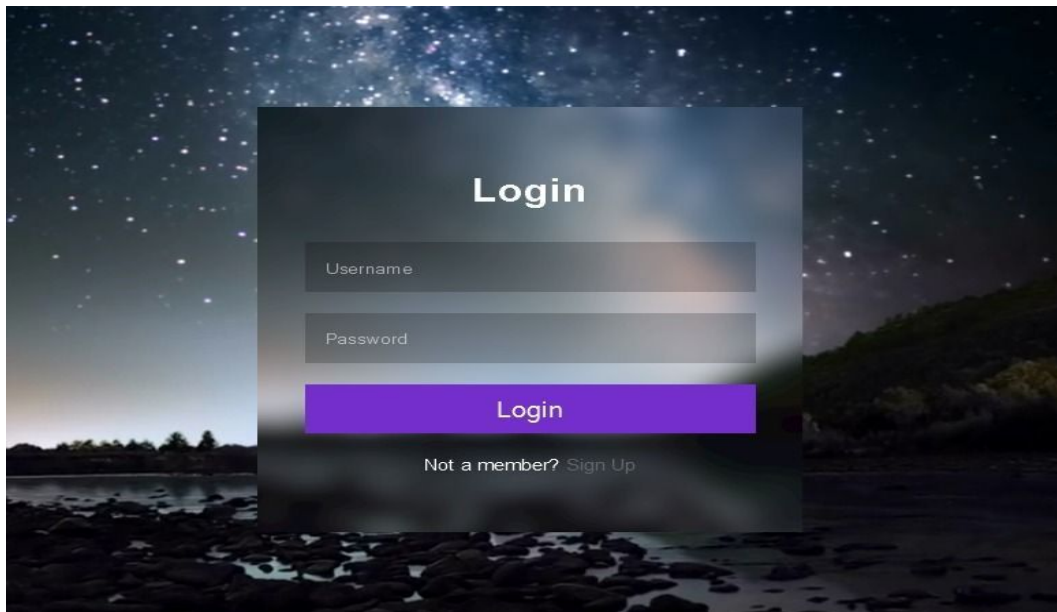


Fig.3.4

3.1.3 BOOTSTRAP

Bootstrap is a sleek, intuitive, and powerful, mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript.

Bootstrap was developed by *Mark Otto* and *Jacob Thornton* at *Twitter*. It was released as an open source product in August 2011 on GitHub.

- Mobile first approach – Bootstrap 3, framework consists of Mobile first styles throughout the entire library instead of in separate files.

- Browser Support – It is supported by all popular browsers.



- Easy to get started – With just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also the Bootstrap official site has good documentation.
- Responsive design – Bootstrap's responsive CSS adjusts to Desktops, Tablets and Mobiles. More about the responsive design is in the chapter Bootstrap Responsive Design.
- Provides a clean and uniform solution for building an interface for developers.
- It contains beautiful and functional built-in components which are easy to customize.
- It also provides web based customization.
- And best of all it is open source.

What Bootstrap Package Includes?

- Scaffolding – Bootstrap provides a basic structure with Grid System, link styles, and background. This is covered in detail in the section Bootstrap Basic Structure
- CSS – Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system. This is covered in detail in the section Bootstrap with CSS.
- Components – Bootstrap contains over a dozen reusable components built to provide iconography, dropdowns, navigation, alerts, pop-overs, and much more. This is covered in detail in the section Layout Components.
- JavaScript Plugins – Bootstrap contains over a dozen custom jQuery plugins. You can easily include them all, or one by one. This is covered in details in the section Bootstrap Plugins.

- Customize – You can customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

Installing and linking bootstrap to the HTML page:

1. Go to Google and type Bootstrap. The Bootstrap links will open.
2. Click the first link (<http://www.getbootstrap.com>) in the search results.
3. The bootstrap homepage will be opened.
4. Click on the "Download Bootstrap" button.
5. The page will be opened to select the Bootstrap option (Complete and minified CSS, JavaScript, fonts, no docs or original source files are included).
6. The file will be downloaded in zip format. You should extract the files.
6. Inside the file, there are three kinds of file available (CSS, FONTS, JS).



fig.3.5

7. Go to Google and type jquery.com
8. The site will be opened.
9. Click the first option and the jQuery page will be opened.

10. Click the "Download jQuery" option.

11. The jQuery file will be opened.

12. There are a number of download links available. Select the advanced version to download.

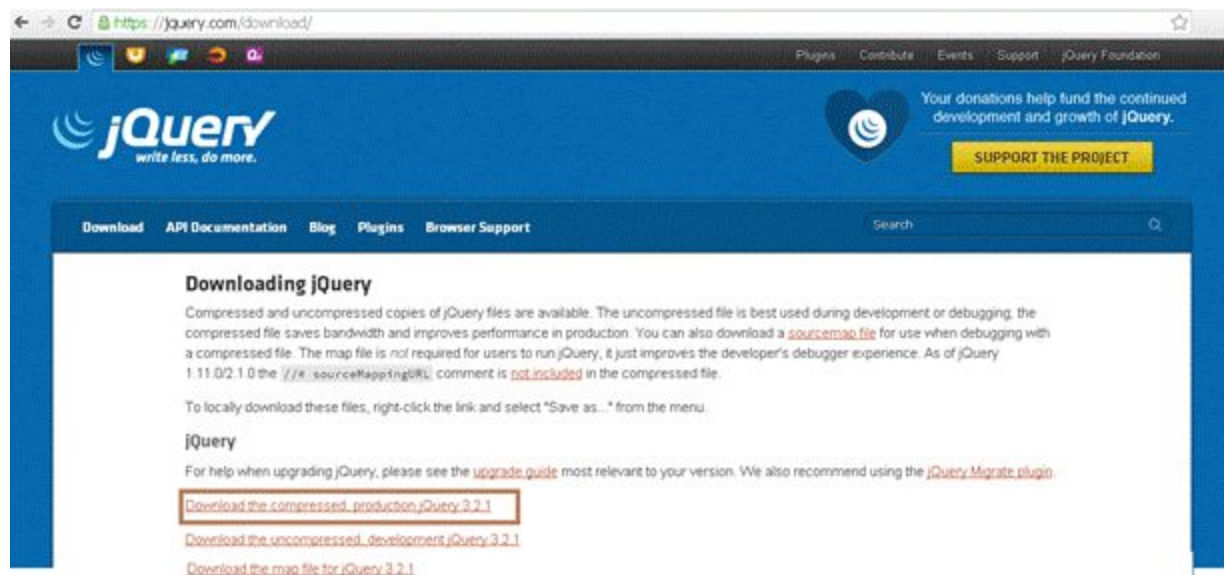
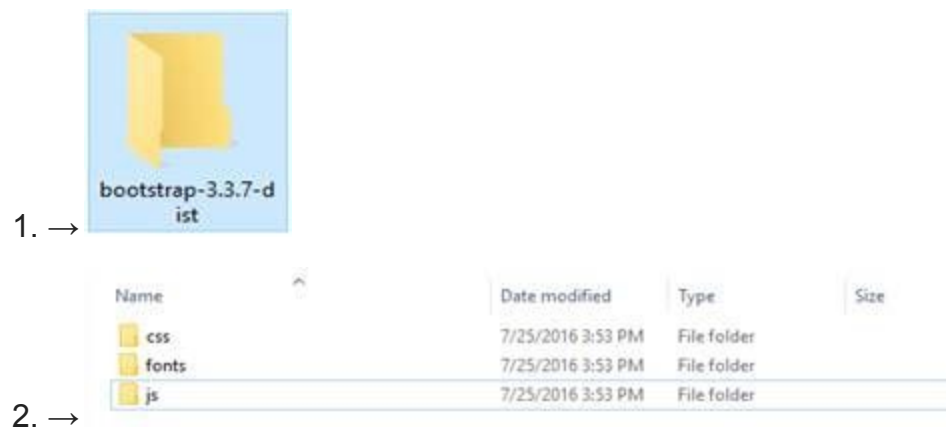


fig.3.6

13. The file will be downloaded. In that file, copy and paste the JS folder (downloaded file folder (bootstrap-3.3.7-dist→js→paste the file)).



3. →

Name	Date modified	Type	Size
css	7/25/2016 3:53 PM	File folder	
fonts	7/25/2016 3:53 PM	File folder	
js	7/25/2016 3:53 PM	File folder	

4. →

Name	Date modified	Type	Size
bootstrap.js	7/25/2016 3:53 PM	JavaScript File	69 KB
bootstrap.min.js	7/25/2016 3:53 PM	JavaScript File	37 KB
jquery.js	11/28/2015 7:15 PM	JavaScript File	94 KB
npm.js	7/25/2016 3:53 PM	JavaScript File	1 KB

14. That is the way to include jquery.js files in the Bootstrap folder.

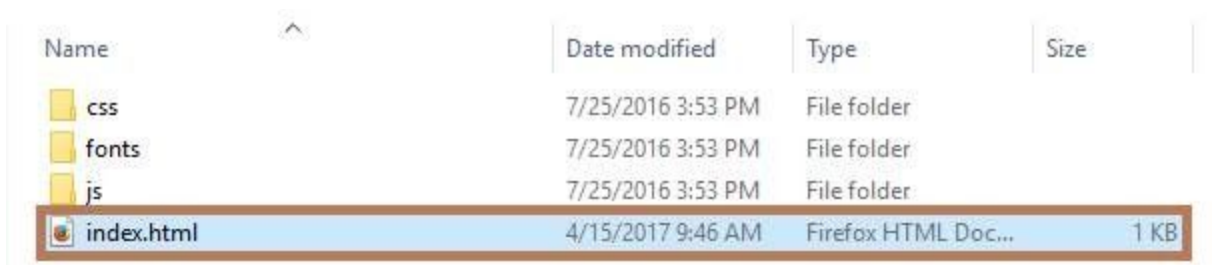
15. After saving the file wherever you want, open the Notepad or Notepad++ (advanced version of Notepad).

16. Write HTML codes for linking all the files in your HTML page.

17. Write the following code.

```
• <!DOCTYPE html>
• <html>
•
• <head>
•   <title>Bootstrap Installation</title>
•   <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css"> </head>
•
• <body>
•   <h1>To link the Bootstrap files and JQuery Files to HTML page
•   <h1>
•       <script type="text/javascript" src="js/bootstrap.min.js"></script>
•       <script type="text/javascript" src="js/jquery.js"></script>
• </body>
•
• </html>
```

18. After writing the codes, it must be saved where the Bootstrap file, for ex- index.html.



Name	Date modified	Type	Size
css	7/25/2016 3:53 PM	File folder	
fonts	7/25/2016 3:53 PM	File folder	
js	7/25/2016 3:53 PM	File folder	
index.html	4/15/2017 9:46 AM	Firefox HTML Doc...	1 KB

18. After saving the file, you should double click the file and the HTML page will run in the browser.

OUTPUT



fig.3.7

3.2 SCRIPTING

All scripting languages are programming languages. The scripting language is basically a language where instructions are written for a run time environment. They do not require the compilation step and are rather interpreted. It brings new functions to applications and glues complex systems together. A scripting language is a programming language designed for integrating and communicating with other programming languages.

There are many scripting languages some of them are discussed below:

- **bash:** It is a scripting language to work in the Linux interface. It is a lot easier to use bash to create scripts than other programming languages. It describes the tools to use and code in the command line and create useful reusable scripts and conserve documentation for other people to work with.
- **Node js:** It is a framework to write network applications using **JavaScript**. Corporate users of Node.js include IBM, LinkedIn, Microsoft, Netflix, PayPal, Yahoo for real-time web applications.
- **Ruby:** There are a lot of reasons to learn the Ruby programming language. Ruby's flexibility has allowed developers to create innovative software. It is a scripting language which is great for web development.
- **Python:** It is easy, free and open source. It supports procedure-oriented programming and object-oriented programming. Python is an interpreted language with dynamic semantics and huge lines of code are scripted and is currently the most hyped language among developers.
- **Perl:** A scripting language with innovative features to make it different and popular. Found on all windows and Linux servers. It helps in text manipulation tasks. High traffic websites that use Perl extensively include priceline.com, IMDB.

Types of scripting languages:

1.Server side scripting languages

2.client side scripting language

Server-side scripting languages run on a web server. When a client sends a request, the server responds by sending content via HTTP

Languages of server side scripting languages;

1.php

2.ASP.NET(C# OR Visual basic)

3.C++

4.java and JSP

5.Python

6.Ruby on Rails and so on.

Popularity of the programming languages in 2020:

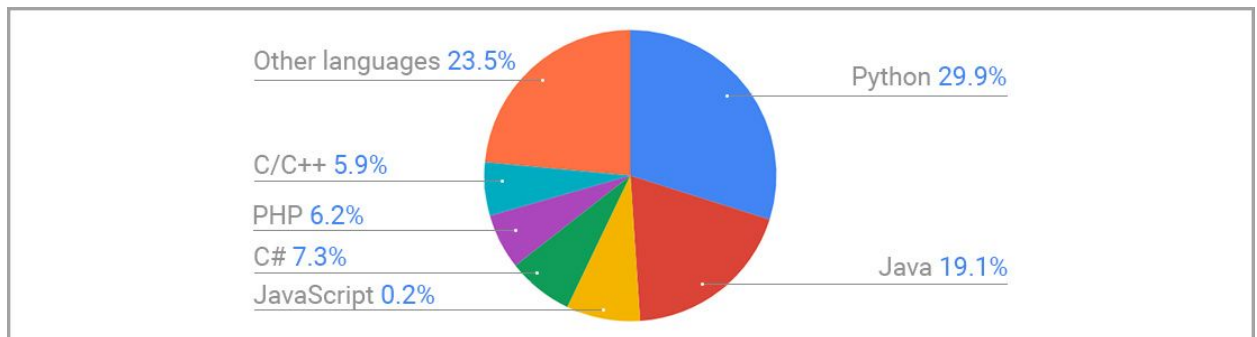


fig.3.8

Client-side scripting languages run on the client end—on their web browser. The benefit of client-side scripts is that they can reduce demand on the server, allowing web pages to load faster. Whereas, one significant benefit of server-side scripts is they are not viewable by the public like client-side scripts are. When trying to decide which way to go on a project, keep in mind that client-side scripting is more focused on user interface and functionality. Conversely, server-side scripting focuses on faster processing, access to data, and resolving errors.

Languages in client side scripting languages:

- JavaScript
- VBScript
- HTML (Structure)
- CSS (Designing)

- AJAX
- jQuery etc.

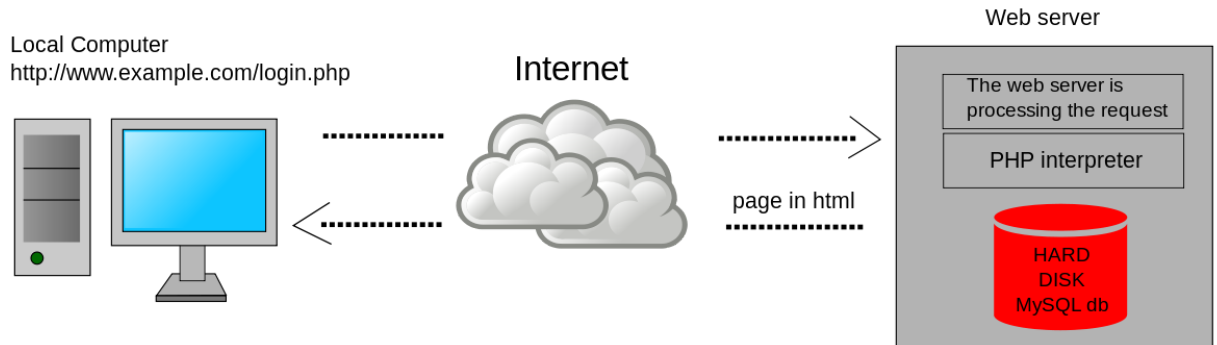


fig.3.9

3.3 DATABASE

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just databases.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.

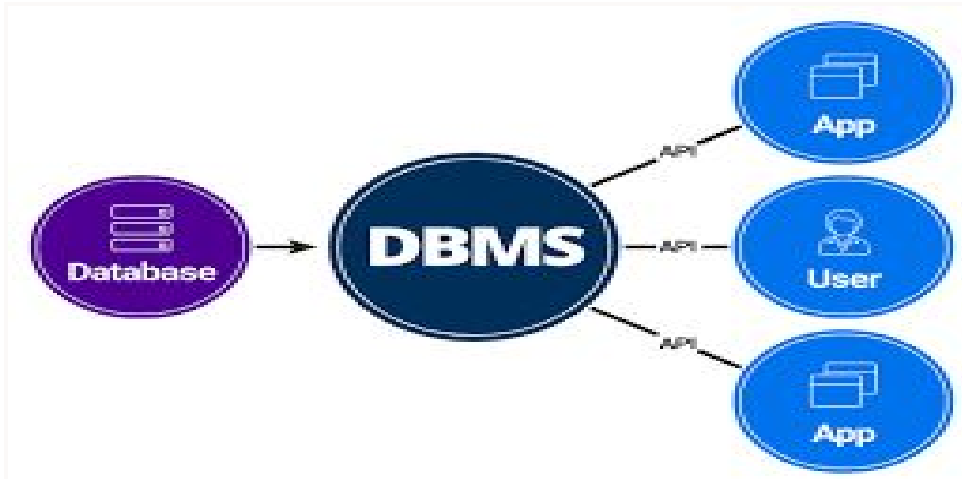


fig.3.10

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data. DBMS allows users to create their own databases as per their requirement. The term "DBMS" includes the user of the database and other application programs. It provides an interface between the data and the software application.

3.3.1.SQL

SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language. This tutorial will give you a quick start to SQL. It covers most of the topics required for a basic understanding of SQL and to get a feel of how it works. SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database. SQL is the standard language for the Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as –

1. MS SQL Server using T-SQL,
2. Oracle using PL/SQL,
3. MS Access version of SQL is called JET SQL (native format) etc.

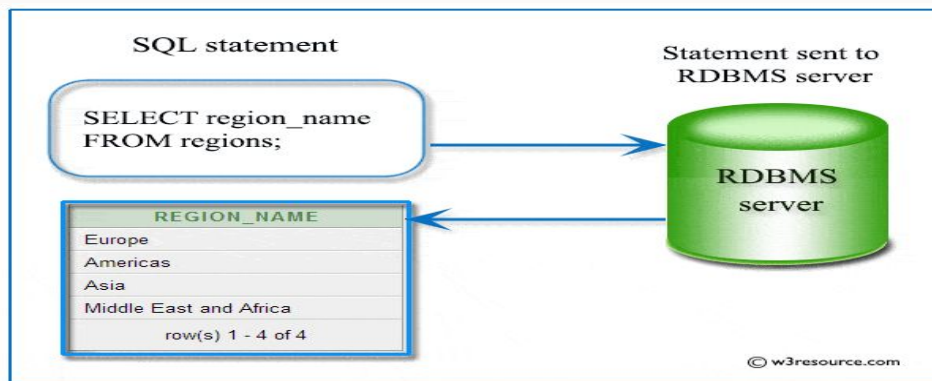


fig.3.11

3.3.2.QUERY

A query is a question, often expressed in a formal way. A database query can be either a select query or an action query. A select query is a data retrieval query, while an action query asks for additional operations on the data, such as insertion, updating or deletion.

Query languages are used to make queries in a database, and Microsoft Structured Query Language (SQL) is the standard. Under the SQL query umbrella, there are several extensions of the language, including MySQL, Oracle SQL and NuoDB.

Query languages for other types of databases, such as NoSQL databases and graph databases, include Cassandra Query Language (CQL), Neo4j's Cypher, Data Mining Extensions (DMX) and XQuery.

Queries can accomplish a few different tasks. Primarily, queries are used to find specific data by filtering specific criteria. Queries can also calculate or summarize data, as well as automate data management tasks. Other queries include parameter, totals, crosstab, make table, append, update and delete. For example, a parameter query runs variations of a particular query, which prompts a user to insert a field value, and then it uses that value to create the criteria, while totals queries allow users to group and summarize data.

Different commands in sql query:

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database

- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index
- WHERE-used to select particular column or row

Still there are more commands in sql query.

CHAPTER-4

SCRIPTING LANGUAGES

Here we have used some scripting languages in the project for web development.

1.Java script

2.JQuery

4.1 JAVASCRIPT

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The ECMA-262 Specification defined a standard version of the core JavaScript language.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform
- Client-Side JavaScript:

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid email address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

4.2 J Query

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License. As of May 2019, jQuery is used by 73% of the 10 million most popular websites. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin, having 3 to 4 times more usage than any other JavaScript library.

jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

The set of jQuery core features—DOM element selections, traversal and manipulation—enabled by its *selector engine* (named "Sizzle" from v1.3), created a new "programming style", fusing algorithms and DOM data structures. This style influenced the architecture of other JavaScript frameworks like YUI v3 and Dojo, later stimulating the creation of the standard *Selectors API*. Later, this style has been enhanced with a deeper algorithm-data fusion in an heir of jQuery, the D3.js framework.

Microsoft and Nokia bundle jQuery on their platforms. Microsoft includes it with Visual Studio for use within Microsoft's ASP.NET AJAX and ASP.NET MV.

CHAPTER-5

REQUIREMENTS

5.1 SOFTWARE REQUIREMENTS:

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

NUMBER	DESCRIPTION
1.Operating system	Windows 7 or later
2.Languages used	HTML,CSS,javascript,SQL,database Jquery,Bootstrap,Ajax.
3.compiler	Visual studio code

table.5.1

5.2 Hardware requirements:

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

Processor	Description
RAM	4.0GB(minimum)
Hard disk drive	400 GB(minimum)

table.5.2

CHAPTER-6

ABOUT PROJECT

6.1 Project Name:MEDICINO(hospital management)

Our project name is MEDICINO which is on hospital management. which provides separate portals for doctors, patients,nurse,receptionists and admin.every portal consists of profile updation,attendance updation and dashboard except in admin and patient portal.patient portal consists of patient information page and admin portal consists of profile updation and reports regarding patients.

6.2 Technologies used:

- 1.HTML
- 2.CSS
- 3.SQL
- 4.Database
- 5.Bootstrap
- 6.Javascript
- 7.J Query
- 8.Ajax

6.3 Technical details:

1. HTML,CSS ,JS,JQuery,Bootstrap are used to design the front end part.
2. SQL,database,Queries are used to develop backend part.
3. Visual studio code is used for compilation.

CHAPTER-7 MODULES

7.1 User registration:

The registration page is a special page where users can create an account on your site. The theme extends the registration form with social networks registration .registration page is for new users of the hospital.

Our registration page consists of following fields:

- 1.email-id
- 2.Username
- 3.Password
- 4.Date of birth
- 5.Mobile number
- 6.Address

These fields are done with validations by using regular expressions

Email-id:

```
var em = /^[a-zA-Z0-9_\-\.]+\@([a-zA-Z0-9_\-\.]+\.[a-zA-Z]{2,5})$/;
```

User name:

```
var us = /^[A-Z][a-z A-Z]{6,12}$/;
```

password:

```
var pas = /^[a-z A-Z 0-9 ! @ # $%]{6,12}$/;
```

Date of birth:

```
var dob = /^(0?[1-9]|[12][0-9]|3[01])([A-Z](0?[1-9]|1[012])([A-Z]\d{4})$/;
```

Mobile number:

```
var num = /^[6-9][0-9]{9}$/;
```

7.2 Login Page:

The login page allows a user to gain access to an application by entering their username and password after registration.

Login page consists of following fields:

- 1.Username
- 2.Password

These are done with validations by using regular expressions

User name:

```
var us = /^[A-Z][a-z A-Z]{6,12}$/;
```

password:

```
var pas = /^[a-z A-Z 0-9 ! @ # $]{6,12}$/;
```

CHAPTER-8

WORKFLOW

A Workflow is a sequence of tasks that processes a set of data. Workflows occur across every kind of business and industry. Anytime data is passed between humans and/or systems, a workflow is created. Workflows are the paths that describe how something goes from being undone to done, or raw to processed.

Workflows happen throughout the workspace. Some are very structured, and others are unstructured, but workflows exist anytime data moves from one task to another.

Here are three major types of workflow:

1. Process Workflow
2. Case Workflow
3. Project Workflow

Process Workflow:

A process workflow happens when the set of tasks is predictable and repetitive. This means that before an item begins the workflow, you know exactly what path it should take. Business process workflows are set up to handle an unlimited number of items going through them. An example is a purchase requisition approval workflow. As soon as it starts, the workflow is set with few variations, and you can process any number of items in a single workflow.

Case Workflow:

In a case workflow, you don't know the path required to complete the item at the start. The path reveals itself as more data is gathered. Support tickets and insurance claims are good examples of cases. It's not clear right from the start how these items will be processed; only after some investigation will the path reveal itself. Similar to process workflows, case workflows can handle any number of items, although they are dependent on a human or an intelligent bot to discern the right path.

Project Workflow:

Projects have a structured path similar to processes, but there may be more flexibility along the way. Think about releasing a new version of your website. You can predict with good accuracy the sequence of tasks required to complete the project. However, project workflow is only good for one item. Another website release may not be done for a long time and will not likely follow exactly the same path.

Most resources you'll find online will only refer to workflows in the sense of process workflow, but the other two are just as important to consider as much of the work around the office falls into those two categories.

Workflow for hospital management:

In hospital management workflow we have 6 classes. they are doctor, patient, receptionist, nurse and admin. Each and every class has their respective attributes with their data types.

For example if we take a doctor class it has attributes like doctor id, name, dept, specialization and their data types are id-int and remaining attributes have string data type.

Workflow diagram of hospital management:

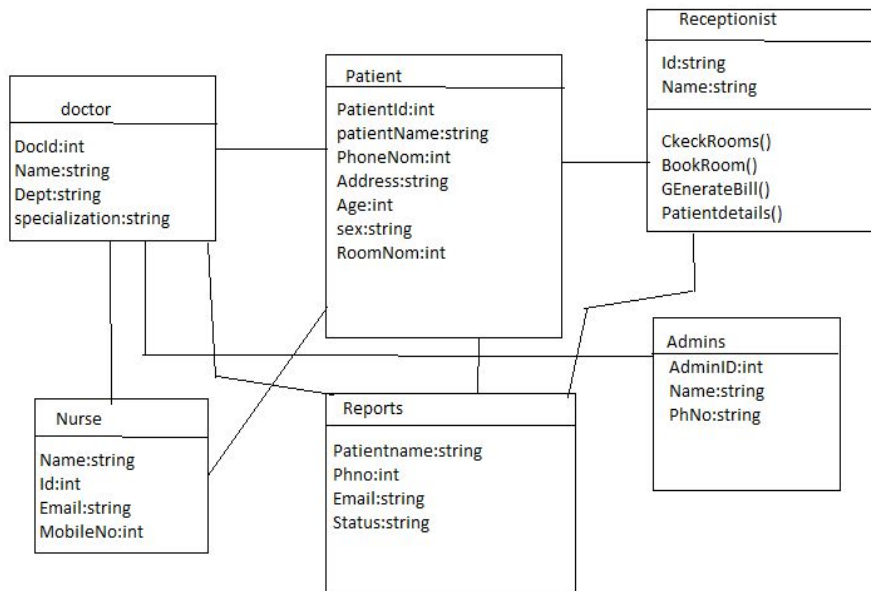


Fig.8.1

CHAPTER-9

OUTPUTS AND SOURCE CODE

FRONTEND CODES AND OUTPUTS:

Registration page:

```
<!DOCTYPE html>
<html>
  <head>
    <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function() {
        $("#btnLogin").click(function() {
          var val1 = $("#txtUsername").val();
          var val2 = $("#txtEmail").val();
          var val3 = $("#txtNumber").val();
          var val4 = $("#txtPassword").val();
          var val5 = $("#txtAddress").val();
          var us = /^[A-Z][a-z A-Z]{6,12}$/;
          var em =
/^[([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+\.[a-zA-Z]{2,5})$/;
          var num = /^[6-9][0-9]{9}$/;
          var pas = /^[a-z A-Z 0-9 ! @ # $]{6,12}$/;

          if(val1==' ' && val2==' ' && val3==' ' && val4==' ' &&
val5==' '){
            $('input[type="text"]').css("border","2px solid
red");
```



```

        $('#input[type="number"]').css("border","2px solid
red");

        $('#input[type="password"]').css("border","2px
solid red");

        window.alert('Please enter all fields');
    }
    else{

        if(us.test(val1) && em.test(val2) &&
num.test(val3) && pas.test(val4)){

            window.alert('Valid');
            var api_url = "http://localhost:3002/login";

            var data = {
                username: $("#txtUsername").val(),
                email: $("#txtEmail").val(),
                mobilenum: $("#txtNumber").val(),
                address:$("#txtAddress").val(),
                password: $("#txtPassword").val(),

            }
            $.ajax({
                url: api_url,
                type: "POST",
                dataType: "Json",
                data: data,
                success: function (data) {
                    return;
                },
                error: function () {
                },
            });

        }
        else{
            window.alert('Invalid');
        }
    }
});

```

```

    });

</script>
<style>
    h1{
        text-align: center;
        color: grey;
        font-family: serif;
        font-style: italic;;
    }
    body{
        background-image:
url("https://www.mountelizabeth.com.sg/Sitefinity/WebSiteTemplates/Parkway
Template/App_Themes/ParkwayTheme/Images/en/background/average-bill-size.pn
g");

        background-size: 100%;
    }
</style>
</head>
<body>
    <h1>CREATE A NEW ACCOUNT</h1>
    <div class="container">
        <form action="" method="POST" id="validateLogin">
            <div class="form-group">

                <label for="Username">Username</label>
                <input type="text" class="form-control"
id="txtUsername" placeholder="Enter Username" />
            </div>
            <div class="form-group">
                <label for="Email">Email</label>
                <input type="text" class="form-control" id="txtEmail"
placeholder="Enter Email" />
            </div>

            <div class="form-group">
                <label for="Mobile Number">Mobile Number</label>

```

```
        <input type="number" class="form-control"
id="txtNumber" placeholder="Enter Mobile Number" />
    </div>
    <div class="form-group">
        <label for="Password">Password</label>
        <input type="password" class="form-control"
id="txtPassword" placeholder="Enter Password" />
    </div>
    <div class="form-group">
        <label
for="Address"><Address>Address</Address></label>
        <textarea rows="2" cols="5" class="form-control"
id="txtAddress" placeholder="Enter Address"></textarea>
    </div>

    <button type="button" class="btn btn-primary"
id="btnLogin">Create</button>
</form>
</div>
</body>
```

OUTPUT:

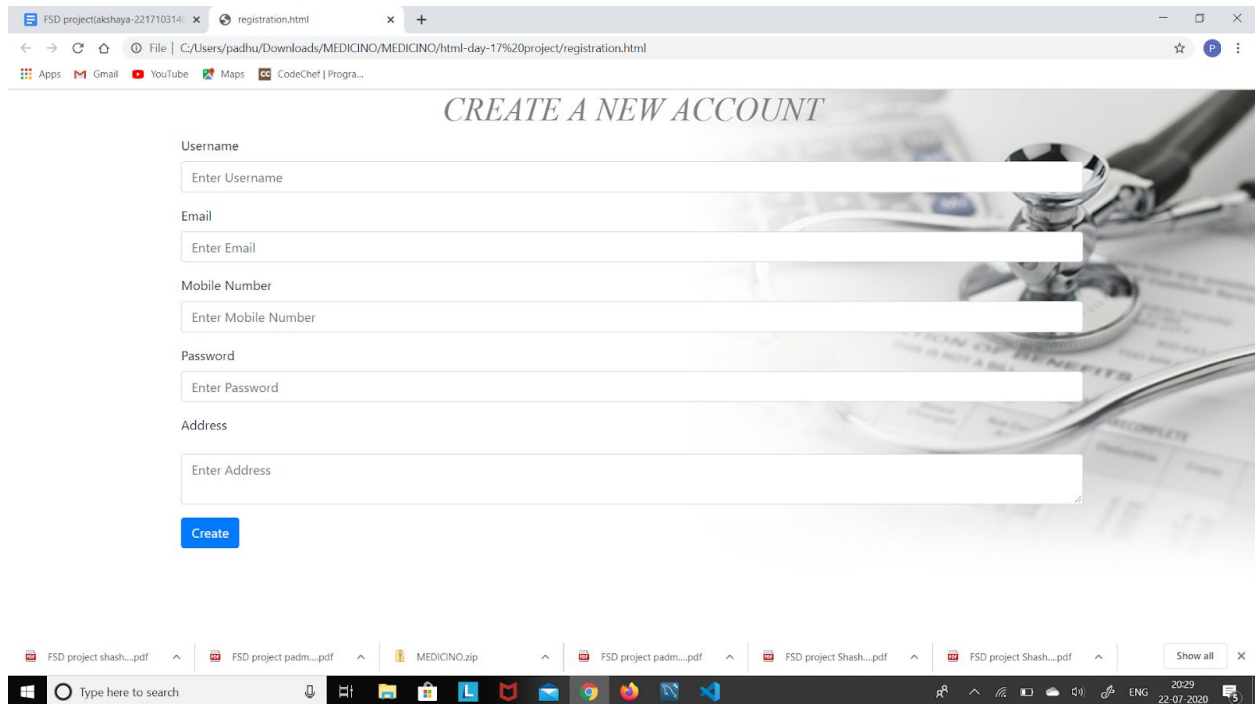


Fig.9.1

Login page:

```
<!DOCTYPE html>
<html>
  <head>
    <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function() {
        $("#btnLogin").click(function()
        {
          var val1 = $("#login-name").val();
          var val2 = $("#login-pass").val();

          var reg=/^[a-z A-Z]{6,12}$/;
```

```

var
reg1=/^((?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[\W]).{6,20})$/;
    if(val1 == '' && val2 == ''){
        $('input[type="text"]').css("border","2px solid
red");

        window.alert('Please enter email and password');

    }
    else{
        if(reg.test(val1) && reg1.test(val2)){
            window.alert('valid');
        }
        else{
            window.alert('Invalid');
        }
    }
});
});
</script>
<style>
    * {
box-sizing: border-box;
    }

    *:focus {
        outline: none;
    }

    body {
font-family: Arial;
background-image:
url("http://getwallpapers.com/wallpaper/full/5/a/b/693674-top-hospital-wallpapers-1920x1080-full-hd.jpg");
padding: 50px;
background-size: 100%;
background-repeat: no-repeat;
    }

```

```
.login {
margin: 20px auto;
width: 300px;
}

.login-screen {
background-color: #FFF;
padding: 20px;
border-radius: 5px
}

.app-title {
text-align: center;
color: #777;
}

.login-form {
text-align: center;
}

.control-group {
margin-bottom: 10px;
}

input {
text-align: center;
background-color: #ECF0F1;
border: 2px solid transparent;
border-radius: 3px;
font-size: 16px;
font-weight: 200;
padding: 10px 0;
width: 250px;
transition: border .5s;
}

input:focus {
border: 2px solid #3498DB;
box-shadow: none;
```

```

}

.btn {
  border: 2px solid transparent;
  background: #3498DB;
  color: #ffffff;
  font-size: 16px;
  line-height: 25px;
  padding: 10px 0;
  text-decoration: none;
  text-shadow: none;
  border-radius: 3px;
  box-shadow: none;
  transition: 0.25s;
  display: block;
  width: 250px;
  margin: 0 auto;
}

.btn:hover {
  background-color: #2980B9;
}

.login-link {
  font-size: 12px;
  color: #444;
  display: block;
  margin-top: 12px;
}
</style>
</head>
<body>
  <div class="login">
    <div class="login-screen">
      <div class="app-title">
        <h1>Login</h1>
      </div>

```

```
        <div class="login-form">
            <div class="control-group">
                <input type="text" class="login-field" value=""
placeholder="username" id="login-name">
                <label class="login-field-icon fui-user"
for="login-name"></label>
            </div>

            <div class="control-group">
                <input type="password" class="login-field" value=""
placeholder="password" id="login-pass">
                <label class="login-field-icon fui-lock"
for="login-pass"></label>
            </div>

            <a class="btn btn-primary btn-large btn-block"
id="btnLogin" href="hospital.html">login</a>
            <a class="login-link" href="registration.html">new
user?</a>
        </div>
    </div>
</div>
</body>
</html>
```


Output:

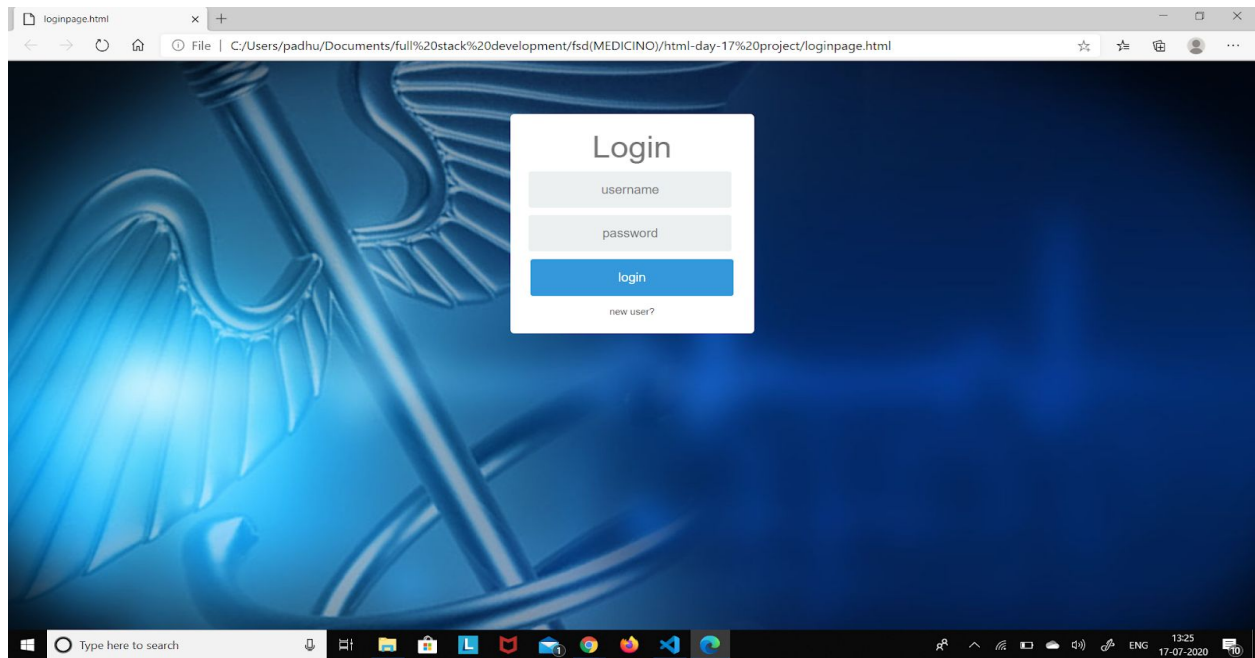


fig.9.2

Medicino home page:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hospital home </title>
    <link rel="stylesheet"
href="bootstrap-4.3.1-dist/css/bootstrap.min.css"/>
    <script rel="stylesheet"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <style type="text/css">
</style>
    <style>
      h1{
        text-align: center;
        font-weight: 600;
        font-style:normal;
      }
      html,body{
```

```
        background-image:
url("https://static-news.moneycontrol.com/static-mcnews/2017/12/Shalby_Hos
pital-770x433.jpg");
        background-size: cover;
        height: 100%;
        padding: 10px;
    }
    h1{
        text-align: center;
        font-family:'Times New Roman', Times, serif;
        color: black;
        font-style:oblique;

    }
    h2,h3{
        color: black;
        text-align:left ;
        font-variant: small-caps;
        font-style: italic;
        font-family:serif;
    }
    h5{
        text-align: right;
        color: darkgreen;
        font-style: italic;

    }

    a{
        font-size: x-large;
        font-style: italic;
        font-family: serif;
    }
    #a2{
        text-align: right;
    }
    .button4 {
```

```

        font-size: 20px;
        background-color: lightcyan;
        border: none;
    }

    li{

        font-weight: lighter;
        font-style: italic;
        background-color: lightcyan;
    }
</style>
</head>
<body>
    <h1>MEDICINO</h1><br/><br/><br/>
    <nav class="navbar navbar-expand-lg navbar ">
        <div class="container">
            <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbar10">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="navbar-collapse collapse" id="navbar10">
                <ul class="navbar-nav nav-fill w-100">
                    <li class="nav-item active">
                        <a class="nav-link font-weight-bold"
href="doctor.html">DOCTOR</a></li>
                    <li class="nav-item">
                        <a class="nav-link font-weight-bold"
href="receptionist.html">RECEPTIONIST </a></li>
                    <li class="nav-item">
                        <a class="nav-link font-weight-bold"
href="patient.html">PATIENT</a></li>
                    <li class="nav-item">
                        <a class="nav-link font-weight-bold"
href="nurse.html">NURSE</a></li>
                    <li class="nav-item">

```

```

        <a class="nav-link font-weight-bold"
href="admin.html">ADMINS</a></li>
    </ul>
</div>
</div>
</nav><br/><br/>
<h2>We are medicino </h2>
<h3>A Medical clinic </h3><br/><br/><br/><br/>

<h5>join us for better health and a happy life</h5><br/>
<div id="a2" class="col-mid-4">
    <button class="button button4" ><a href="appointment.html">
Appointment</a></button><br><br>
    <button class="button button4 " ><a href="loginpage.html"> Logout</a></button>
</div>
</body>
</html>

```

Output:

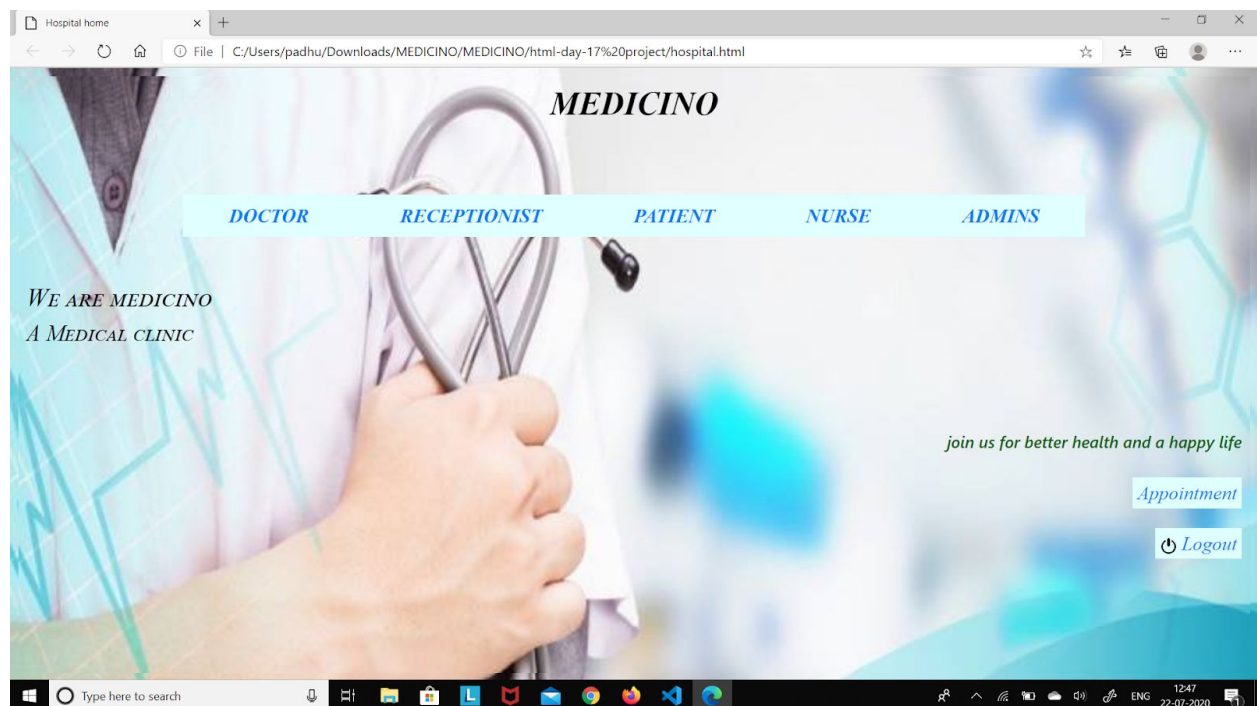


Fig.9.3

Doctor portal:

```

<!DOCTYPE html>
<html>
  <head>
    <title>Hospital home </title>
    <link rel="stylesheet"
href="bootstrap-4.3.1-dist/css/bootstrap.min.css"/>
    <script rel="stylesheet"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <style type="text/css">
    </style>
    <style>
      h1{
        text-align: center;
        font-weight: 600;
        font-style:normal;

      }
      html,body{
        background-image:
url("https://static-news.moneycontrol.com/static-mcnews/2017/12/Shalby_Hos
pital-770x433.jpg");
        background-size: cover;
        height: 100%;
        padding: 10px;
      }
      h1{
        text-align: center;
        font-family:'Times New Roman', Times, serif;
        color: black;
        font-style:oblique;

      }
      h2,h3{
        color: black;
        text-align:left ;
        font-variant: small-caps;
        font-style: italic;

```



```

        <a class="nav-link font-weight-bold"
href="doctor.html">DOCTOR</a></li>
        <li class="nav-item">
            <a class="nav-link font-weight-bold"
href="receptionist.html">RECEPTIONIST </a></li>
        <li class="nav-item">
            <a class="nav-link font-weight-bold"
href="patient.html">PATIENT</a></li>
        <li class="nav-item">
            <a class="nav-link font-weight-bold"
href="nurse.html">NURSE</a></li>
        <li class="nav-item">
            <a class="nav-link font-weight-bold"
href="admin.html">ADMINS</a></li>
    </ul>
</div>
</div>
</nav><br/><br/>
<h2>We are medicino </h2>
<h3>A Medical clinic </h3><br/><br/><br/><br/>

<h5>join us for better health and a happy life</h5><br/>
<div id="a2" class="col-mid-4">
    <button type="button" class="btn btn-info"><a
href="appointment.html"> Appointment</a></button>
</div>
</body>
</html>

```

Output:

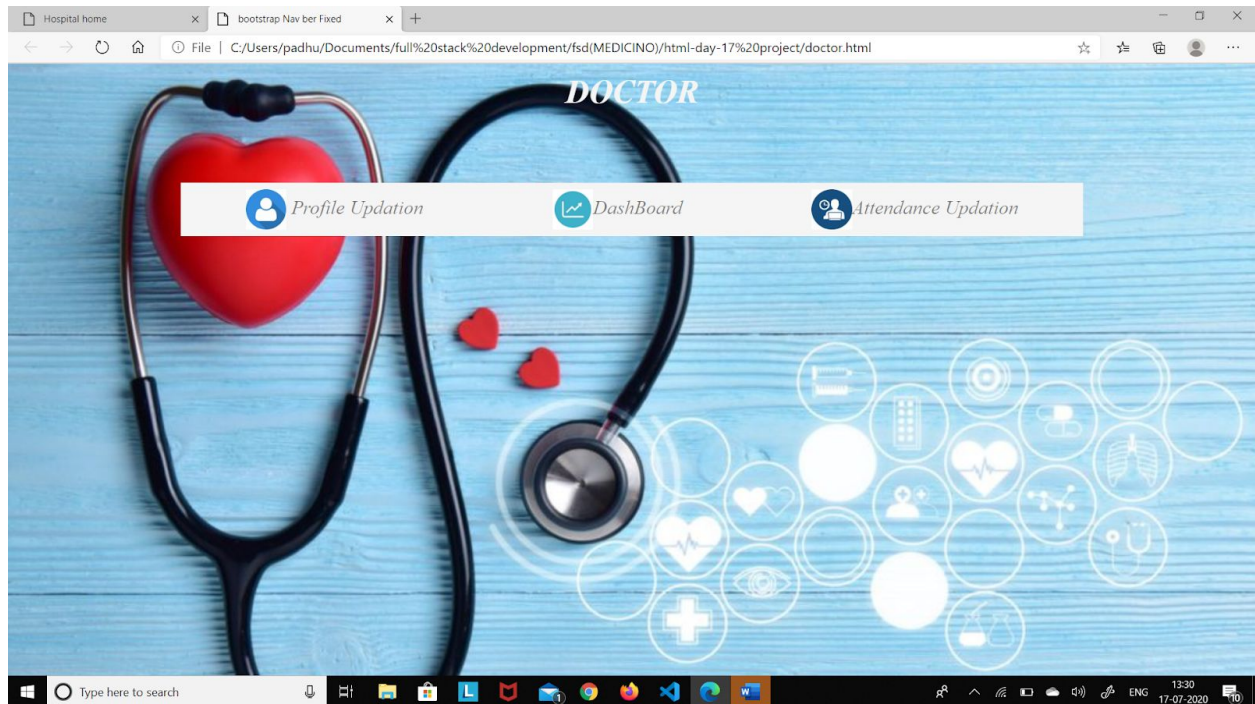


Fig.9.4

Nurse portal:

```
<!DOCTYPE html>
<html>
  <head>
    <title>bootstrap Nav ber Fixed</title>
    <link rel="stylesheet"
href="bootstrap-4.3.1-dist/css/bootstrap.min.css"/>
    <script rel="stylesheet"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <style type="text/css">
      body{
background-image:url ("http://i.huffpost.com/gen/3709316/images/o-STUDENT-N
URSE-facebook.jpg") ;
        background-repeat: no-repeat;
        padding: 10px;
      }
      img{
        height: 50px;
        width: 50px;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-md-12">
          <div class="card">
            <div class="card-header">
              <h3>DOCTOR</h3>
            </div>
            <div class="card-body">
              <div class="d-flex justify-content-between">
                <a href="#">Profile Updation</a>
                <a href="#">DashBoard</a>
                <a href="#">Attendance Updation</a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```



```

a{
    background-color: whitesmoke;
    font-size: x-large;
    font-family: serif;
    font-style: italic;
}
h1{
    text-align: center;
    font-weight: bold;
    color: whitesmoke;
    font-family: serif;
    font-style: italic;
}
</style>
</head>
<body>
    <h1>NURSE</h1><br/><br/><br/>
    <nav class="navbar navbar-expand-lg navbar-light ">
        <div class="container">
            <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbar10">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="navbar-collapse collapse" id="navbar10">
                <ul class="navbar-nav nav-fill w-100">
                    <li class="nav-item"><a class="nav-link"
href="profile.html"> Profile Updation</a></li>
                    <li class="nav-item"><a class="nav-link"
href="dashboard-nurse.html">DashBoard</a></li>
                    <li class="nav-item"><a class="nav-link"
href="attendance.html">Attendance Updation</a></li>

```

```

        </ul>
      </div>
    </div>
  </nav>

</body>
</html>

```

Output:

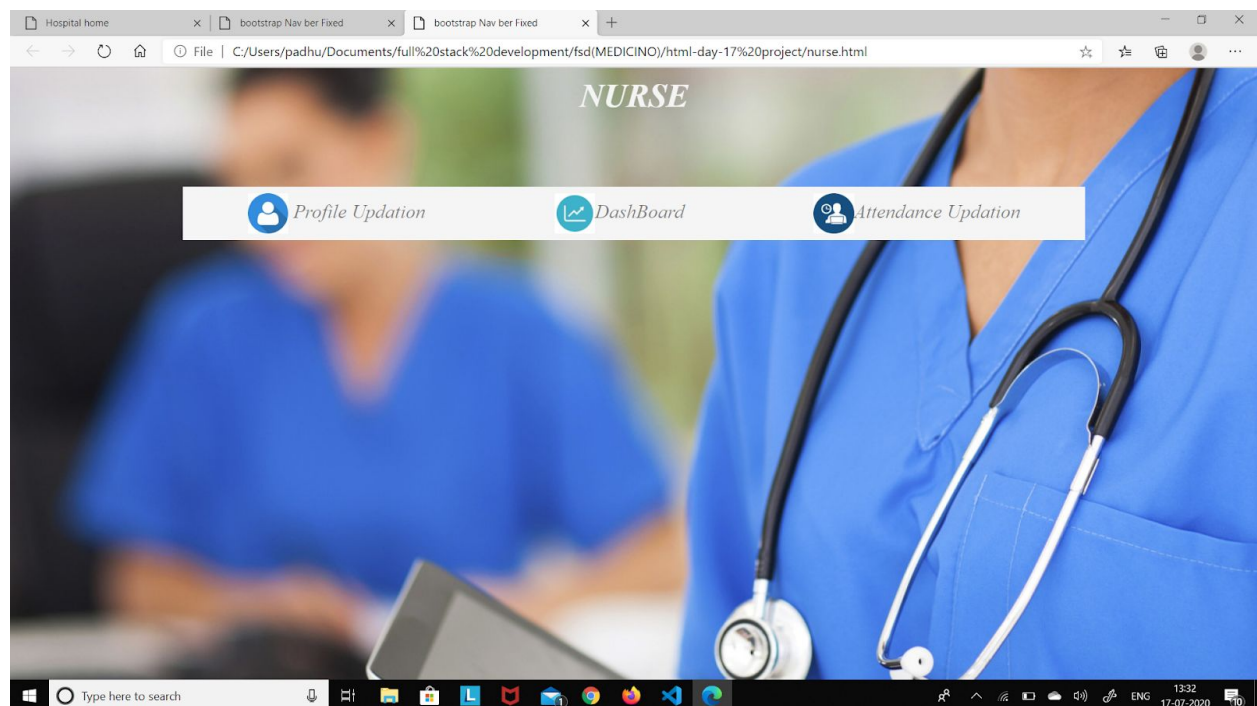


fig.9.5

Receptionist portal:

```

<!DOCTYPE html>

<html>

  <head>

```

```

<title>bootstrap Nav ber Fixed</title>
<link rel="stylesheet"
href="bootstrap-4.3.1-dist/css/bootstrap.min.css"/>
<script rel="stylesheet"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
<style type="text/css">
  body{

background-image:url("https://img.rawpixel.com/s3fs-private/rawpixel_image
s/website_content/440-kaboompics-3009.jpg?w=800&dpr=1&fit=default&crop=def
ault&auto=format&fm=pjpg&q=75&vib=3&con=3&usm=15&bg=F4F4F3&ixlib=js-2.2.1&
s=6c12a7cab9e7024bbb01bb6ab7ca7a8e") ;
    background-repeat: no-repeat;
    background-size: 100%;
    padding: 10px;
  }
  img{
    height: 50px;
    width: 50px;
  }
  a{
    background-color: whitesmoke;
    font-size: x-large;
    font-family: serif;
    font-style: italic;
  }
  h1{
    text-align: center;
    font-weight: bold;
    color: whitesmoke;
    font-family: serif;
    font-style: italic;
  }
</style>
</head>
<body>
  <h1>RECEPTIONIST</h1><br/><br/><br/>

```

```

<nav class="navbar navbar-expand-lg navbar-light ">
  <div class="container">
    <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbar10">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="navbar-collapse collapse" id="navbar10">
      <ul class="navbar-nav nav-fill w-100">
        <li class="nav-item">
          <a class="nav-link" href="profile.html"> Profile Updation</a></li>
          <li class="nav-item"><a class="nav-link"
href="attendance.html">Attendance Updation</a></li>

        </ul>
      </div>
    </div>
  </nav>
</body>
</html>

```

Output:

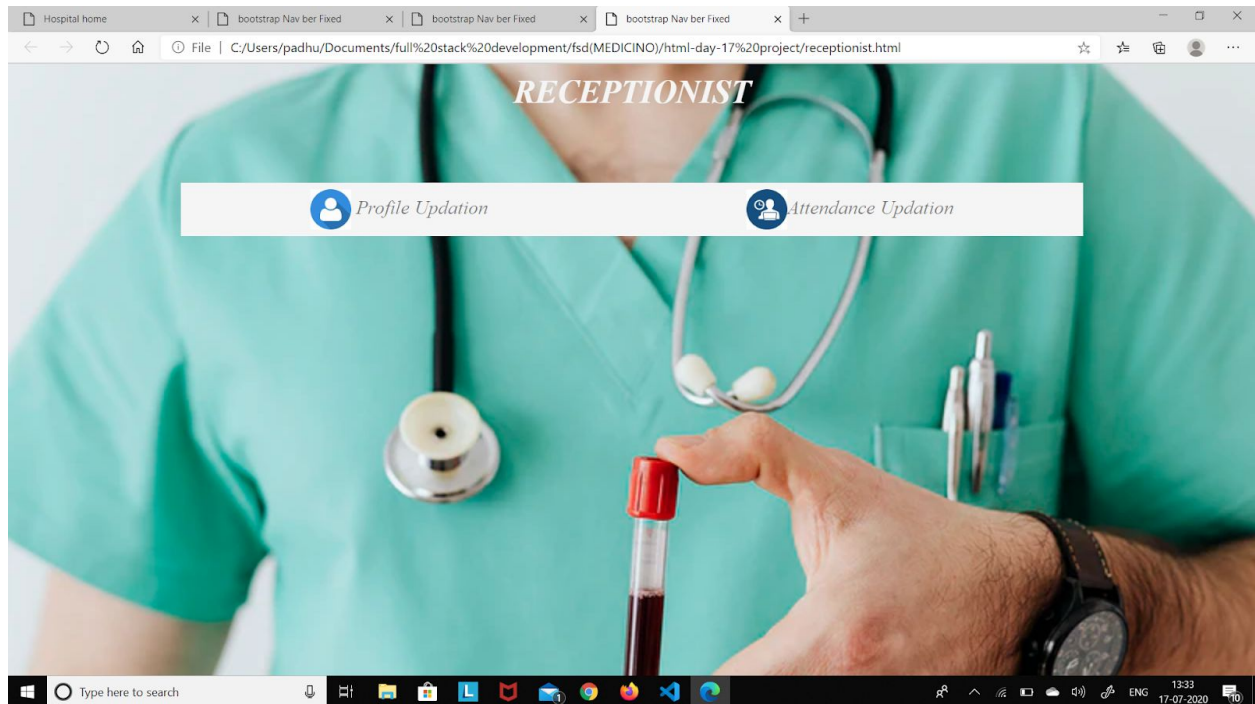


fig.9.6

Admin portal:

```
<!DOCTYPE html>
<html>
  <head>
    <title>bootstrap Nav ber Fixed</title>
    <link rel="stylesheet"
href="bootstrap-4.3.1-dist/css/bootstrap.min.css"/>
    <script rel="stylesheet"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <style type="text/css">
      body{
background-image:url ("https://www.som.com/FILE/25189/nslij_1575x900_eduard
hueber_archphoto_updated_01jpg.jpg") ;
        background-repeat: no-repeat;
        background-size: 100%;
        padding: 10px;
      }
      img{
        height: 50px;
        width: 50px;
```

```

    }
    a{
        background-color: whitesmoke;
        font-size: x-large;
        font-family: serif;
        font-style: italic;
    }
    h1{
        text-align: center;
        font-weight: bold;
        color: white;
        font-family: serif;
        font-style: italic;
    }
</style>
</head>
<body>
    <h1>ADMINS</h1><br/><br/><br/>
    <nav class="navbar navbar-expand-lg navbar-light ">
        <div class="container">
            <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbar10">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="navbar-collapse collapse" id="navbar10">
                <ul class="navbar-nav nav-fill w-100">
                    <li class="nav-item"><a class="nav-link"
href="profile.html"> PROFILE </a></li>
                    <li class="nav-item"><a class="nav-link"
href="reports.html">REPORTS</a></li>

                </ul>
            </div>
        </div>
    </nav>

```

```

        </div>
      </nav>

    </body>
  </html>

```

Output:

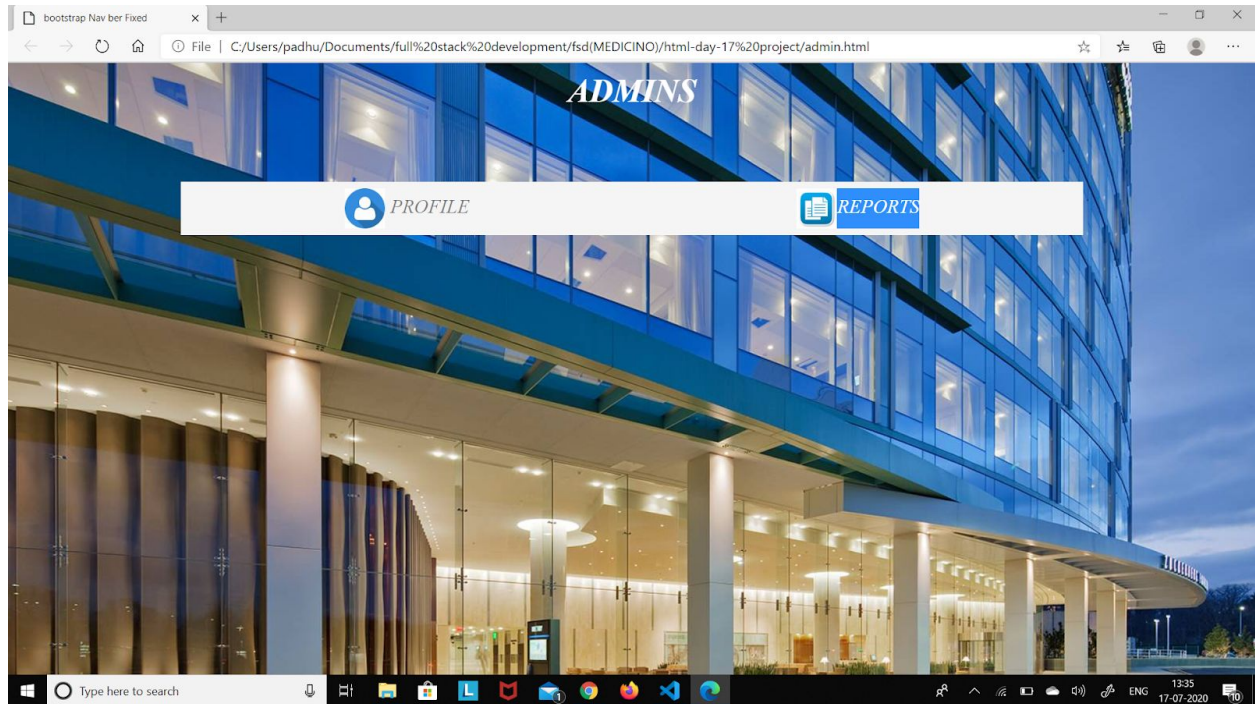


fig.9.7

Patient portal:

```

<!DOCTYPE html>
<html>
  <head>
    <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function() {
        $("#btnLogin").click(function() {

```

```

        var val1 = $("#txtname").val();
        var val2 = $("#txtEmail").val();
        var val3 = $("#txtNumber").val();
        var val4 = $("#txtPassword").val();
        var val5 = $("#txtAddress").val();
        var us = /^[a-z A-Z]{6,12}$/;
        var em = /^[a-z A-Z 0-9 . -
_] {4,8} [a-z]{3,5} [a-z]{2,4}$/;
        var num = /^[6-9][0-9]{9}$/;
        var pas = /^[a-z A-Z 0-9 ! @ # $]{6,12}$/;
        var add = /^[a-z A-Z 0-9]{6,12}$/;
        if(val1==' ' && val2==' ' && val3==' ' && val4==' ' &&
val5==' '){
            $('#input[type="text"]').css("border","2px solid
red");
            $('#input[type="number"]').css("border","2px solid
red");
            window.alert('Please enter all fields');
        }
        else{
            if(us.test(val1) && em.test(val2) &&
num.test(val3) && pas.test(val4) && add.test(val5)){
                window.alert('Valid');
            }
            else{
                window.alert('Invalid');
            }
        }
    });
});

</script>
<style>
    h1{
        text-align: center;
        color: black;

```



```

        font-family: serif;
        font-style: italic;
    }
    body{
        background-image:
url("https://associatesinmedtox.com/wp-content/uploads/2016/12/lvkwE1.jpg"
);

        color: black;

    }

</style>
</head>
<body>
    <h1>PATIENT INFORMATION </h1>
    <div class="container">
        <form action="" method="POST" id="validateLogin">
            <div class="form-group">
                <label for="name">Name</label>
                <input type="text" class="form-control" id="txtname"
placeholder="Enter name" />
            </div>
            <div class="form-group">
                <label for="Email">Email</label>
                <input type="text" class="form-control" id="txtEmail"
placeholder="Enter Email" />
            </div>
            <div class="form-group">
                <label for="Mobile Number">Mobile Number</label>
                <input type="number" class="form-control"
id="txtNumber" placeholder="Enter Mobile Number" />
            </div>
            <div class="form-group">
                <label for="Password">Password</label>
                <input type="text" class="form-control"
id="txtPassword" placeholder="Enter Password" />
            </div>
        </form>
    </div>

```

```

        <div class="form-group">
            <label
for="Address"><Address>Address</Address></label>
            <textarea rows="5" cols="10" class="form-control"
id="txtAddress" placeholder="Enter Address"></textarea>
        </div>
        <button type="button" class="btn btn-primary"
id="btnLogin">Update</button>
    </form>
</div>
</body>

```

Output:

The screenshot shows a web browser window with the address bar displaying the file path: C:/Users/padhu/Documents/full%20stack%20development/fsd(MEDICINO)/html-day-17%20project/patient.html. The page content is titled "PATIENT INFORMATION" in a serif font. Below the title, there are five input fields with labels: "Name" (with placeholder "Enter name"), "Email" (with placeholder "Enter Email"), "Mobile Number" (with placeholder "Enter Mobile Number"), "Password" (with placeholder "Enter Password"), and "Address" (with placeholder "Enter Address"). At the bottom of the form is a blue "Update" button. The background of the page is a blue-tinted image of medical equipment and a person in a lab coat. The Windows taskbar is visible at the bottom of the browser window.

Fig.9.8

Profile updation pages for doctor ,nurse ,receptionist and admin:

Admin profile code:

```

<!DOCTYPE html>
<html>
    <head>
        <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
        <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
        <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>

```

```

<script type="text/javascript">
    $(document).ready(function() {
        $("#btnLogin").click(function() {
            var val1 = $("#txtUsername").val();
            var val2 = $("#txtEmail").val();
            var val3 = $("#txtNumber").val();
            var val4 = $("#txtPassword").val();
            var val5 = $("#txtAddress").val();
            var us = /^[a-z A-Z]{6,12}$/;
            var em =
/^[([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+\.[a-zA-Z]{2,5})$/;
            var num = /^[6-9][0-9]{9}$/;
            var pas = /^[a-z A-Z 0-9 ! @ # $]{6,12}$/;

            if(val1==' ' && val2==' ' && val3==' ' && val4==' ' &&
val5==' '){
                $('input[type="text"]').css("border","2px solid
red");
                $('input[type="number"]').css("border","2px solid
red");

                window.alert('Please enter all fields');
            }
            else{

                if(us.test(val1) && em.test(val2) &&
num.test(val3) && pas.test(val4)){
                    window.alert('Valid');
                    var api_url = "http://localhost:3002/admins";

                    var data = {
                        username: $("#txtUsername").val(),
                        emailid: $("#txtEmail").val(),
                        mobilenum: $("#txtNumber").val(),
                        password1: $("#txtPassword").val(),
                        address:$("#txtAddress").val(),
                    }
                    $.ajax({
                        url: api_url,

```

```

        type: "POST",
        dataType: "Json",
        data: data,
        success: function (data) {
            return;
        },
        error: function () {
        },
    });

    }
    else{
        window.alert('Invalid');
    }
}
});
});

</script>
<style>
    body{
background-image:url("https://associatesinmedtox.com/wp-content/uploads/20
16/12/lvkwE1.jpg");
        color: black;
    }
</style>
</head>
<body>
    <div class="container">
        <form action="" method="POST" id="validateLogin">
            <div class="form-group">
                <label for="Username">Username</label>
                <input type="text" class="form-control"
id="txtUsername" placeholder="Enter Username" />
            </div>
            <div class="form-group">

```

```

        <label for="Email">Email</label>
        <input type="text" class="form-control" id="txtEmail"
placeholder="Enter Email" />
    </div>
    <div class="form-group">
        <label for="Mobile Number">Mobile Number</label>
        <input type="number" class="form-control"
id="txtNumber" placeholder="Enter Mobile Number" />
    </div>
    <div class="form-group">
        <label for="Password">Password</label>
        <input type="password" class="form-control"
id="txtPassword" placeholder="Enter Password" />
    </div>
    <div class="form-group">
        <label
for="Address"><Address>Address</Address></label>
        <textarea rows="5" cols="10" class="form-control"
id="txtAddress" placeholder="Enter Address"></textarea>
    </div>
    <button type="button" class="btn btn-primary"
id="btnLogin">Update</button>
</form>
</div>
</body>

```

Nurse profile page:

```

<!DOCTYPE html>
<html>
    <head>
        <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
        <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
        <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>
        <script type="text/javascript">
            $(document).ready(function() {

```

```

        $("#btnLogin").click(function(){
            var val1 = $("#txtUsername").val();
            var val2 = $("#txtEmail").val();
            var val3 = $("#txtNumber").val();
            var val4 = $("#txtPassword").val();
            var val5 = $("#txtAddress").val();
            var us = /^[a-zA-Z]{6,12}$/;
            var em =
/^[a-zA-Z0-9_\-\.]+\@([a-zA-Z0-9_\-\.]+\.[a-zA-Z]{2,5})$/;
            var num = /^[6-9][0-9]{9}$/;
            var pas = /^[a-zA-Z 0-9 ! @ # $]{6,12}$/;

            if(val1==' ' && val2==' ' && val3==' ' && val4==' ' &&
val5==' '){

                $('input[type="text"]').css("border","2px solid
red");

                $('input[type="number"]').css("border","2px solid
red");

                window.alert('Please enter all fields');
            }
            else{

                if(us.test(val1) && em.test(val2) &&
num.test(val3) && pas.test(val4)){

                    window.alert('Valid');
                    var api_url = "http://localhost:3002/nurses";
                    var data = {
                        username: $("#txtUsername").val(),
                        emailid: $("#txtEmail").val(),
                        mobilenum: $("#txtNumber").val(),
                        password1: $("#txtPassword").val(),
                        address:$("#txtAddress").val(),
                    }
                    $.ajax({
                        url: api_url,
                        type: "POST",
                        dataType: "Json",

```

```

        data: data,
        success: function (data) {
            return;
        },
        error: function () {
        },
    });

        }
        else{
            window.alert('Invalid');
        }
    }
    });
});

</script>
<style>
    body{

background-image:url("https://associatesinmedtox.com/wp-content/uploads/20
16/12/lvkwE1.jpg");

        color: black;

    }
</style>
</head>
<body>
    <div class="container">
        <form action="" method="POST" id="validateLogin">
            <div class="form-group">
                <label for="Username">Username</label>
                <input type="text" class="form-control"
id="txtUsername" placeholder="Enter Username" />
            </div>
            <div class="form-group">
                <label for="Email">Email</label>

```

```

        <input type="text" class="form-control" id="txtEmail"
placeholder="Enter Email" />
    </div>
    <div class="form-group">
        <label for="Mobile Number">Mobile Number</label>
        <input type="number" class="form-control"
id="txtNumber" placeholder="Enter Mobile Number" />
    </div>
    <div class="form-group">
        <label for="Password">Password</label>
        <input type="password" class="form-control"
id="txtPassword" placeholder="Enter Password" />
    </div>
    <div class="form-group">
        <label
for="Address"><Address>Address</Address></label>
        <textarea rows="5" cols="10" class="form-control"
id="txtAddress" placeholder="Enter Address"></textarea>
    </div>
    <button type="button" class="btn btn-primary"
id="btnLogin">Update</button>
</form>
</div>
</body>

```

Receptionist profile page:

```

<!DOCTYPE html>
<html>
    <head>
        <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
        <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
        <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>
        <script type="text/javascript">
            $(document).ready(function() {

```



```

        $("#btnLogin").click(function(){
            var val1 = $("#txtUsername").val();
            var val2 = $("#txtEmail").val();
            var val3 = $("#txtNumber").val();
            var val4 = $("#txtPassword").val();
            var val5 = $("#txtAddress").val();
            var us = /^[a-z A-Z]{6,12}$/;
            var em =
/^[([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+\.[a-zA-Z]{2,5})$/;
            var num = /^[6-9][0-9]{9}$/;
            var pas = /^[a-z A-Z 0-9 ! @ # $]{6,12}$/;

            if(val1==' ' && val2==' ' && val3==' ' && val4==' ' &&
val5==' '){

                $('input[type="text"]').css("border","2px solid
red");

                $('input[type="number"]').css("border","2px solid
red");

                window.alert('Please enter all fields');
            }
            else{

                if(us.test(val1) && em.test(val2) &&
num.test(val3) && pas.test(val4)){
                    window.alert('Valid');
                    var api_url =
"http://localhost:3002/reception";
                    var data = {
                        username: $("#txtUsername").val(),
                        emailid: $("#txtEmail").val(),
                        mobilenumber: $("#txtNumber").val(),
                        password1: $("#txtPassword").val(),
                        address:$("#txtAddress").val(),
                    }
                    $.ajax({
                        url: api_url,
                        type: "POST",

```

```

        dataType: "Json",
        data: data,
        success: function (data) {
            return;
        },
        error: function () {
        },
    });

    }
    else{
        window.alert('Invalid');
    }
    }
    });
});

</script>
<style>
    body{
background-image:url("https://associatesinmedtox.com/wp-content/uploads/20
16/12/lvkwE1.jpg");
        color: black;
    }
</style>
</head>
<body>
    <div class="container">
        <form action="" method="POST" id="validateLogin">
            <div class="form-group">
                <label for="Username">Username</label>
                <input type="text" class="form-control"
id="txtUsername" placeholder="Enter Username" />
            </div>
            <div class="form-group">
                <label for="Email">Email</label>

```

```
        <input type="text" class="form-control" id="txtEmail"
placeholder="Enter Email" />
    </div>
    <div class="form-group">
        <label for="Mobile Number">Mobile Number</label>
        <input type="number" class="form-control"
id="txtNumber" placeholder="Enter Mobile Number" />
    </div>
    <div class="form-group">
        <label for="Password">Password</label>
        <input type="password" class="form-control"
id="txtPassword" placeholder="Enter Password" />
    </div>
    <div class="form-group">
        <label
for="Address"><Address>Address</Address></label>
        <textarea rows="5" cols="10" class="form-control"
id="txtAddress" placeholder="Enter Address"></textarea>
    </div>
    <button type="button" class="btn btn-primary"
id="btnLogin">Update</button>
</form>
</div>
</body>
```

Output:

Fig.9.9

Attendance updation page:

```
<!DOCTYPE html>
<html>
  <head>
    <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function() {
        $("#btnLogin").click(function()
        {
          var val1 = $("#txtname").val();
          var val2 = $("#txtfeedback").val();

          var reg=/^[A-Z][a-z A-Z]{10,12}$/;
```

```

        if(val1 == '' && val2 == ''){
            $('#input[type="text"]').css("border","2px solid
red");

            window.alert('Please enter name and feedback');

        }
        else{
            if(reg.test(val1)){
                window.alert('valid');
                var api_url =
"http://localhost:3002/attendance";
                var data = {
                    name: $("#txtname").val(),
                    feedback: $("#txtfeedback").val(),
                }
                $.ajax({
                    url: api_url,
                    type: "POST",
                    dataType: "Json",
                    data: data,
                    success: function (data) {
                        return;
                    },
                    error: function () {
                    },
                });
            }
            else{
                window.alert('Invalid');
            }
        }
    });
});
</script>
<style>
h1{

```

```

        text-align: center;
    }
    body{
        background-image:
url("https://www.neeyamo.com/assets/images/2017/12/Time-Attendance-Managem
ent.png");

        background-repeat: no-repeat;

    }
</style>
</head>
<body>
    <h1>Attendance Updation </h1>
    <div class="container">
        <form action="" method="POST" id="validateLogin">
            <div class="form-group">
                <label for="name">Name</label>
                <input type="text" class="form-control" id="txtname"
placeholder="Enter name" />
            </div>
            <div class="form-group">
                <label for="feedback">Feedback</label>
                <textarea class="form-control" id="txtfeedback" >
</textarea>
            </div>
            <button type="button" class="btn btn-primary"
id="btnLogin" >Update</button>
        </form>
    </div>

</body>
</html>

```

Output:

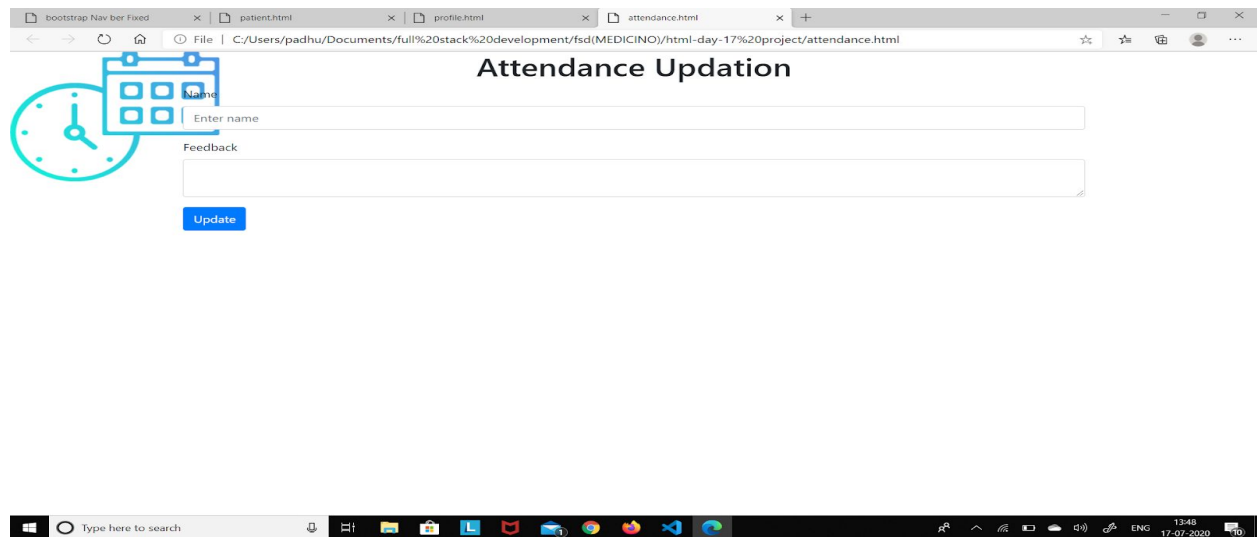


Fig.9.10

Doctor dashboard:

```
<!DOCTYPE html>
<html>
  <head>
    <title>dashboard</title>
    <style>
      h3{
        font-weight: 700;
      }
      .a1{
        border:solid 3px black;
      }
      .a2{
        border:solid 3px black;
      }
    </style>
  </head>
```

```

<body>
  <h3>The doctor to patient graph is as given below</h3>
  
  
</body>
</html>

```

Output:

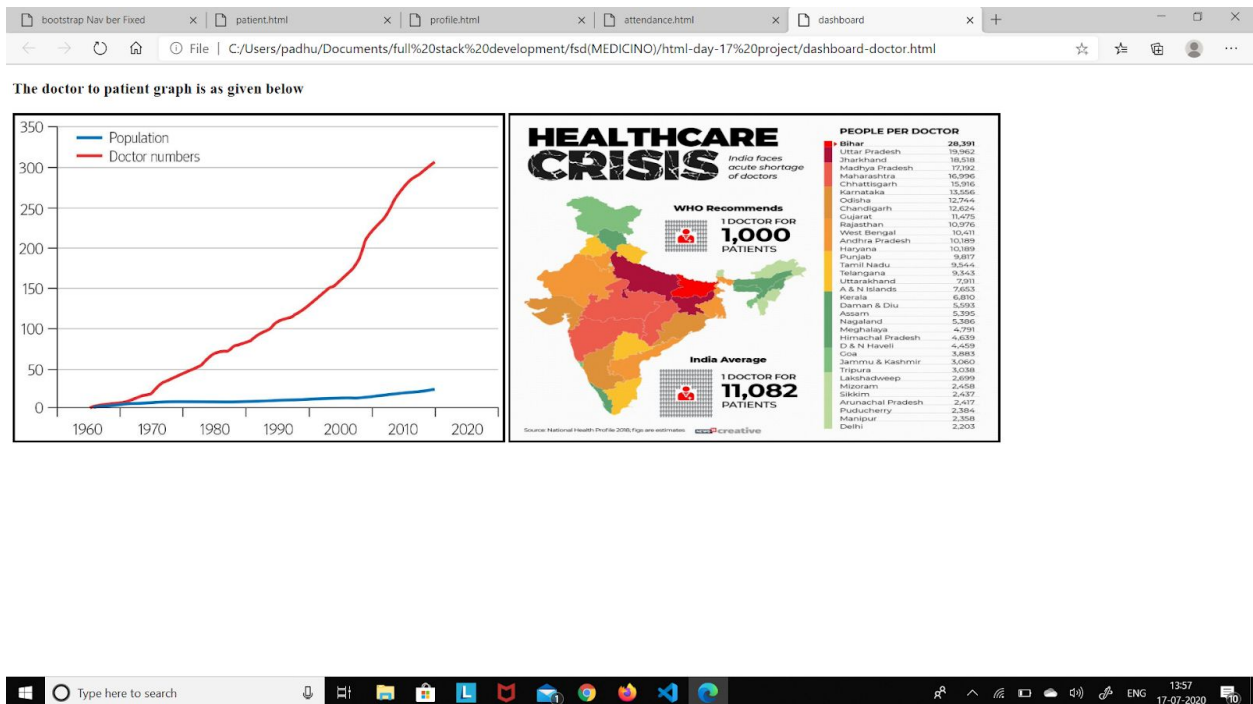


Fig.9.11

Nurse dashboard:

```

<!DOCTYPE html>
<html>
  <head>
    <title>dashboard</title>
    <style>
      h3{
        font-weight: 700;
      }
      img{

```



```

border:solid 3px black;

}

</style>
</head>
<body>
    <h3>The Nurse to patient Ratio is as given below</h3>
    
    <h3>Diagnostics</h3>
    

</body>
</html>

```

Output:

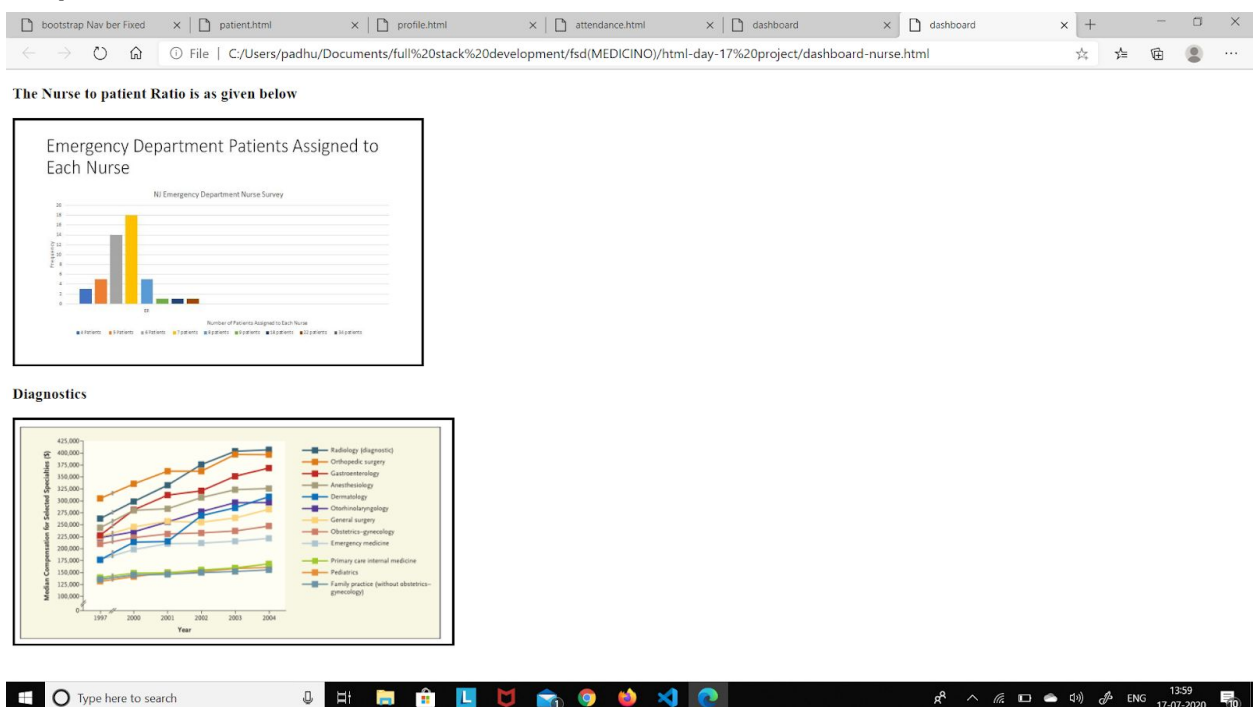


Fig.9.12

Patient report page in admin portal:

```

<!DOCTYPE html>
<html>

```

```

<head>
    <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>
    <script type="text/javascript">
        $(document).ready(function() {
            $("#btnLogin").click(function() {
                var val1 = $("#txtUsername").val();
                var val2 = $("#txtNumber").val();
                var val3 = $("#txtaddress").val();
                var val4 = $("#txtstatus").val();
                var us = /^[a-z A-Z]{6,12}$/;
                var num = /^[6-9][0-9]{9}$/;
                if(val1==' ' && val2==' ' && val3==' ' && val4==' '){
                    $('input[type="text"]').css("border","2px solid
red");
                    $('input[type="number"]').css("border","2px solid
red");
                    window.alert('Please enter all fields');
                }

                if(us.test(val1) && num.test(val2)){
                    window.alert('Valid');
                    var api_url = "http://localhost:3002/reports";
var data = {
    patientname: $("#txtUsername").val(),
    mobilenummer: $("#txtNumber").val(),
    address:$("#txtAddress").val(),
    status:$("#txtAddress1").val(),
    }
$.ajax({
    url: api_url,
    type: "POST",
    dataType: "Json",

```

```

        data: data,
        success: function (data) {
            return;
        },
        error: function () {
        },
    });

    }
    else{
        window.alert('Invalid');
    }

    });
});

</script>
<style>
    body{
        background-image:
url("https://i.pinimg.com/originals/b2/14/42/b214425a7284ce662493da77720b9
410.jpg");

        background-repeat:no-repeat;
        background-size: cover;
        height: 100%;
    }
</style>
</head>
<body>
    <div class="container">
        <form action="" method="POST" id="validateLogin">
            <div class="form-group">
                <label for="Username">Patient's Name</label>
                <input type="text" class="form-control"
id="txtUsername" placeholder="Enter patient name" />
            </div>
            <div class="form-group">
                <label for="Mobile Number">Mobile Number</label>

```

```

        <input type="number" class="form-control"
id="txtNumber" placeholder="Enter Mobile Number" />
    </div>
    <div class="form-group">
        <label
for="Address"><Address>Address</Address></label>
        <textarea rows="5" cols="10" class="form-control"
id="txtAddress" placeholder="Enter Address"></textarea>
    </div>
    <div class="form-group">
        <label for="Address"><Address>Status</Address></label>
        <textarea rows="5" cols="10" class="form-control"
id="txtAddress1" placeholder="Enter enter the status of
treatment"></textarea>
    </div>
    <button type="button" class="btn btn-primary"
id="btnLogin">Update</button>
</form>
</div>
</body>

```

Output:

The screenshot shows a web browser window with multiple tabs. The active tab is 'reports.html'. The browser address bar shows the file path: 'C:/Users/padhu/Documents/full%20stack%20development/fsd(MEDICINO)/html-day-17%20project/reports.html'. The web page displays a form with the following fields:

- Patient's Name:** A text input field with the placeholder 'Enter patient name'.
- Mobile Number:** A text input field with the placeholder 'Enter Mobile Number'.
- Address:** A text area with the placeholder 'Enter Address'.
- Status:** A text area with the placeholder 'Enter enter the status of treatment'.
- Update:** A blue button located at the bottom left of the form.

The background of the form is a blurred image of various colored pills (purple, yellow, blue, pink) on a dark surface.

Fig.9.13

Appointment page:

```
<!DOCTYPE html>
<html>
  <head>
    <link href="bootstrap-4.3.1-dist/css/bootstrap.min.css"
rel="StyleSheet" />
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/bootstrap.min.js"></script>
    <script type="text/javascript"
src="bootstrap-4.3.1-dist/js/jquery.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function() {
        $("#btnLogin").click(function() {
          var val1 = $("#txtname").val();
          var val2 = $("#txtEmail").val();
          var val3 = $("#txtNumber").val();

          var val5 = $("#txtAddress").val();
          var us = /^[a-z A-Z]{6,12}$/;
          var em = /^[a-z A-Z 0-9 . -
_] {4,8} [a-z]{3,5} [a-z]{2,4}$/;
          var num = /^[6-9][0-9]{9}$/;

          var add = /^[a-z A-Z 0-9]{6,12}$/;
          if(val1==' ' && val2==' ' && val3==' ' && val5==' '){
            $('input[type="text"]').css("border","2px solid
red");
            $('input[type="number"]').css("border","2px solid
red");
            window.alert('Please enter all fields');
          }
          else{
            if(us.test(val1) && em.test(val2) &&
num.test(val3) && add.test(val5)){
              window.alert('Valid');
```

```

        var api_url =
"http://localhost:3002/appointment";

        var data = {
            name: $("#txtname").val(),
            email: $("#txtEmail").val(),
            mobilenumber: $("#txtNumber").val(),
            address:$("#txtAddress").val(),
        }
$.ajax({
    url: api_url,
    type: "POST",
    dataType: "Json",
    data: data,
    success: function (data) {
        return;
    },
    error: function () {
    },
});

        }
        else{
            window.alert('Invalid');
        }
    }
});
});

</script>
<style>
    h1{
        text-align: center;
        color: black;
        font-family: serif;
        font-style: italic;
    }
    body{

```

```

        background-image:
url("https://associatesinmedtox.com/wp-content/uploads/2016/12/lvkwE1.jpg"
);

        padding: 10px;
        color: black;

    }

</style>
</head>
<body>
    <h1>PATIENT INFORMATION </h1>
    <div class="container">
        <form action="" method="POST" id="validateLogin">
            <div class="form-group">
                <label for="name">Name</label>
                <input type="text" class="form-control" id="txtname"
placeholder="Enter name" />
            </div>
            <div class="form-group">
                <label for="Email">Email</label>
                <input type="text" class="form-control" id="txtEmail"
placeholder="Enter Email" />
            </div>
            <div class="form-group">
                <label for="Mobile Number">Mobile Number</label>
                <input type="number" class="form-control"
id="txtNumber" placeholder="Enter Mobile Number" />
            </div>
            <div class="form-group">
                <label
for="Address"><Address>Address</Address></label>
                <textarea rows="5" cols="10" class="form-control"
id="txtAddress" placeholder="Enter Address"></textarea>
            </div>
            <button type="button" class="btn btn-primary"
id="btnLogin">Book</button>
        </form>
    </div>

```

```

        </form>

    </div>

</body>

```

Output:

The screenshot shows a web browser window with the URL `File | C:/Users/padhu/Downloads/MEDICINO/MEDICINO/html-day-17%20project/appointment.html`. The page displays a form titled *PATIENT INFORMATION* on a blue background with a medical theme. The form contains the following fields and a button:

- Name:** A text input field with the placeholder text "Enter name".
- Email:** A text input field with the placeholder text "Enter Email".
- Mobile Number:** A text input field with the placeholder text "Enter Mobile Number".
- Address:** A larger text input field with the placeholder text "Enter Address".
- Book:** A blue button with white text.

The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system clock indicates the date is 22-07-2020 and the time is 20:46.

Fig.9.14

BACKEND CODES:

Database code:

```

create database db2;
use db2;
create table registers(
    doctorid int auto_increment,
    username varchar(100),
    emailid varchar(100),
    mobilenum bigint,
    password1 varchar(20),
    address varchar(200),
    primary key(doctorid)
);
select *from registers;
create table reception(
    receptionid int auto_incremen
    username varchar(100),

```



```
    emailid varchar(100),
    mobilenumbr bigint,
    password1 varchar(20),
    address varchar(200),
    primary key(receptionid)
);
create table patient(
    patientid int auto_increment,
    username varchar(100),
    emailid varchar(100),
    mobilenumbr bigint,
    password1 varchar(20),
    address varchar(200),
    primary key(patientid)
);
create table nurses(
    nurseid int auto_increment,
    username varchar(100),
    emailid varchar(100),
    mobilenumbr bigint,
    password1 varchar(20),
    address varchar(200),
    primary key(nurseid)
);
create table admin(
    adminid int auto_increment,
    username varchar(100),
    emailid varchar(100),
    mobilenumbr bigint,
    password1 varchar(20),
    address varchar(200),
    primary key(adminid)
);
```

```
create table reports(
    patientid int auto_increment,
    patientname varchar(30),
    mobilenumbr bigint,
    address varchar(200),
```

```
status varchar(200),
primary key(patientid)
);
create table login(
id int auto_increment,
username varchar(30),
email varchar(100),
mobilenumber bigint,
address varchar(200),
password varchar(20),
primary key(id)
);
select * from nurseupd;
create table recept(
name varchar(30),
feedback varchar(50)
);
create table appointment(
patientid int auto_increment,
name varchar(100),
email varchar(100),
mobilenumber bigint,
address varchar(200),
primary key(patientid)
);
create table updation(
name varchar(30),
feedback varchar(50)
);
create table attendance(
name varchar(30),
feedback varchar(50)
);
create table nurseupd(
name varchar(30),
feedback varchar(50)
);
select * from login;
show tables;
```

```
select * from patient;
select * from attendance;
select * from updation;
select * from nurseupd;
```

Code for config:

```
module.exports = {
  HOST: "localhost",
  USER: "root",
  PASSWORD: "dsps@123",
  DB: "db2",
};
```

Controllers code:

Admin controller code:

```
const Admin = require("../models/admin.model.js");

exports.create = (req, res) => {
  // Validate request
  if (!req.body) {
    res.status(400).send({
      message: "Content can not be empty!"
    });
  }

  const admin = new Admin({
    username: req.body.username,
    emailid: req.body.emailid,
    mobilenum: req.body.mobilenum,
    password1: req.body.password1,
    address: req.body.address,
  });
```

```

Admin.create(admin, (err, data) => {
  if (err)
    res.status(500).send({
      message:
        err.message || "Some error occurred while creating the admin."
    });
  else res.send(data);
});
};

```

Appointment controller code:

```
const Appointment = require("../models/appointment.model.js");
```

```

exports.create = (req, res) => {
  // Validate request
  if (!req.body) {
    res.status(400).send({
      message: "Content can not be empty!"
    });
  }
}

```

```

const appointment = new Appointment({
  name:req.body.name,
  email: req.body.email,
  mobilenumber: req.body.mobilenumber,
  address:req.body.address,
});

```

```

Appointment.create(appointment, (err, data) => {
  if (err)
    res.status(500).send({
      message:

```

```

    err.message || "Some error occurred while creating the Appointment."
  });
  else res.send(data);
});
};

```

Attendance controller code:

```

const Attendanced = require("../models/attendanced.model.js");

exports.create = (req, res) => {
  // Validate request
  if (!req.body) {
    res.status(400).send({
      message: "Content can not be empty!"
    });
  }

  const attendanced = new Attendanced({
    name: req.body.name,
    feedback: req.body.feedback,
  });

  Attendanced.create(attendanced, (err, data) => {
    if (err)
      res.status(500).send({
        message:
          err.message || "Some error occurred while creating the attendance."
      });
    else res.send(data);
  });
};

```

Login controller code:

```
const Logins = require("../models/login.model.js");

exports.create = (req, res) => {
  // Validate request
  if (!req.body) {
    res.status(400).send({
      message: "Content can not be empty!"
    });
  }

  const logins = new Logins({
    username: req.body.username,
    email: req.body.email,
    dateofbirth: req.body.dateofbirth,
    mobilenumber: req.body.mobilenumber,
    address: req.body.address,
    password: req.body.password,
  });

  Logins.create(logins, (err, data) => {
    if (err)
      res.status(500).send({
        message:
          err.message || "Some error occurred while creating the login."
      });
    else res.send(data);
  });
};
```

Nurse controller code:

```

const Nurse = require("../models/nurse.model.js");

exports.create = (req, res) => {
  // Validate request
  if (!req.body) {
    res.status(400).send({
      message: "Content can not be empty!"
    });
  }

  const nurse = new Nurse({
    username: req.body.username,
    emailid: req.body.emailid,
    mobilenum: req.body.mobilenum,
    password1: req.body.password1,
    address: req.body.address,
  });

  Nurse.create(nurse, (err, data) => {
    if (err)
      res.status(500).send({
        message:
          err.message || "Some error occurred while creating the Nurse."
      });
    else res.send(data);
  });
};

```

Patient controller code:

```

const Patient = require("../models/patient.model.js");

exports.create = (req, res) => {

```

```

// Validate request
if (!req.body) {
  res.status(400).send({
    message: "Content can not be empty!"
  });
}

const patient = new Patient({
  username: req.body.username,
  emailid: req.body.emailid,
  mobilenum: req.body.mobilenum,
  password1: req.body.password1,
  address: req.body.address,
});

Patient.create(patient, (err, data) => {
  if (err)
    res.status(500).send({
      message:
        err.message || "Some error occurred while creating the patient."
    });
  else res.send(data);
});
};

```

Reception controller code:

```

const Reception = require("../models/reception.model.js");

exports.create = (req, res) => {
  // Validate request
  if (!req.body) {
    res.status(400).send({

```



```

        message: "Content can not be empty!"
    });
}

const reception = new Reception({
    username:req.body.username,
    emailid: req.body.emailid,
    mobilenumber: req.body.mobilenumber,
    password1: req.body.password1,
    address:req.body.address,
});

Reception.create(reception, (err, data) => {
    if (err)
        res.status(500).send({
            message:
                err.message || "Some error occurred while creating the
reception."
        });
    else res.send(data);
});
};

```

Register controller code:

```

const Register = require("../models/register.model.js");

// Create and Save a new Register
exports.create = (req, res) => {
    // Validate request
    if (!req.body) {
        res.status(400).send({
            message: "Content can not be empty!"

```

```

    });
  }

  // Create a Register
  const register = new Register({
    username:req.body.username,
    emailid: req.body.emailid,
    mobilenummer: req.body.mobilenummer,
    password1: req.body.password1,
    address:req.body.address,
  });

  // Save Register in the database
  Register.create(register, (err, data) => {
    if (err)
      res.status(500).send({
        message:
          err.message || "Some error occurred while creating the Register."
      });
    else res.send(data);
  });
};

```

Reports controller code:

```
const Reports = require("../models/report.model.js");
```

```
exports.create = (req, res) => {
```

```
  // Validate request
```

```
  if (!req.body) {
```

```
    res.status(400).send({
```

```

    message: "Content can not be empty!"
  });
}

const reports = new Reports({
  patientname:req.body.patientname,
  mobilenumber: req.body.mobilenumber,
  address:req.body.address,
  status:req.body.status,
});

Reports.create(reports, (err, data) => {
  if (err)
    res.status(500).send({
      message:
        err.message || "Some error occurred while creating the Reports."
    });
  else res.send(data);
});
};

```

Updation controller:

```
const Updation = require("../models/updation.model.js");
```

```

exports.create = (req, res) => {
  // Validate request
  if (!req.body) {
    res.status(400).send({
      message: "Content can not be empty!"
    });
  }
}

```

```

    const updatation = new Updation({
      name:req.body.name,
      feedback: req.body.feedback,
    });

    Updation.create(updatation, (err, data) => {
      if (err)
        res.status(500).send({
          message:
            err.message || "Some error occurred while creating the updatation."
        });
      else res.send(data);
    });
  };
};

```

MODELS:

Admin:

```

const sql = require('../models/db.js');

const Admin = function (admin) {
  this.username = admin.username;
  this.emailid = admin.emailid;
  this.mobilenumber = admin.mobilenumber;
  this.password1 = admin.password1;
  this.address = admin.address;
};

Admin.create = (newadmin, result) => {
  sql.query(`insert into admin set ?`, newadmin, (err, res) => {
    if (err) {
      console.log(err);
      result(err, null);
    }
  });
};

```

```

        return;
    }

    console.log("Created admin : ", { id: res.insertedId, ...newadmin });
    return (null, { id: res.insertedId, ...newadmin });
  })
};

module.exports = Admin;

```

Appointment:

```

const sql = require('../models/db.js');

const Appointment = function(appointment){
  this.name = appointment.name;
  this.email = appointment.email;
  this.mobilenumber = appointment.mobilenumber;
  this.address=appointment.address;
};

Appointment.create = (newAppointment,result) => {
  sql.query('insert into appointment set ?',newAppointment,(err,res) =>{
    if(err){
      console.log(err);
      result(err,null);
      return;
    }

    console.log("Created appointment : ",{id:res.insertedId,...newAppointment});
    return (null,{id:res.insertedId,...newAppointment});
  })
};

module.exports = Appointment;

```

Attendance:

```

const sql = require('../models/db.js');

```

```

const Attendanced = function(attendanced){
  this.name = attendanced.name;
  this.feedback = attendanced.feedback;
};

Attendanced.create = (newAttendanced,result) => {
  sql.query('insert into attendance set ?',newAttendanced,(err,res) =>{
    if(err){
      console.log(err);
      result(err,null);
      return;
    }
    console.log("Created attendance : ",{id:res.insertedId,...newAttendanced});
    return (null,{id:res.insertedId,...newAttendanced});
  })
};

module.exports = Attendanced;

```

Database:

```

const mysql = require('mysql');
const dbConfig = require('../config/db.config.js');
const connection = mysql.createConnection({
  host: dbConfig.HOST,
  user: dbConfig.USER,
  password: dbConfig.PASSWORD,
  database: dbConfig.DB
});

connection.connect(error =>{

```

```

    if(error){
      return console.error(error.message);
    }
    console.log('Successfully connected to MySQL Database');
  });

module.exports = connection;

```

Login:

```

const sql = require('../models/db.js');

const Logins = function(logins){
  this.username = logins.username;
  this.email = logins.email;
  this.dateofbirth=logins.dateofbirth;
  this.mobilenumber = logins.mobilenumber;
  this.address=logins.address;
  this.password=logins.password;
};

Logins.create = (newLogins,result) => {
  sql.query('insert into login set ?',newLogins,(err,res) =>{
    if(err){
      console.log(err);
      result(err,null);
      return;
    }
    console.log("Created login : ",{id:res.insertedId,...newLogins});
    return (null,{id:res.insertedId,...newLogins});
  })
};

```

```
module.exports = Logins;
```

Nurse:

```
const sql = require('../models/db.js');
```

```
const Nurse = function(nurse){
```

```
  this.username = nurse.username;
```

```
  this.emailid = nurse.emailid;
```

```
  this.mobilenumber = nurse.mobilenumber;
```

```
  this.password1=nurse.password1;
```

```
  this.address=nurse.address;
```

```
};
```

```
Nurse.create = (newNurse,result) => {
```

```
  sql.query('insert into nurses set ?',newNurse,(err,res) =>{
```

```
    if(err){
```

```
      console.log(err);
```

```
      result(err,null);
```

```
      return;
```

```
    }
```

```
    console.log("Created Nurse : ",{id:res.insertedId,...newNurse});
```

```
    return (null,{id:res.insertedId,...newNurse});
```

```
  })
```

```
};
```

```
module.exports = Nurse;
```

Patient:

```
const sql = require('../models/db.js');
```



```

const Patient = function(patient){
  this.username = patient.username;
  this.emailid = patient.emailid;
  this.mobilenumber = patient.mobilenumber;
  this.password1=patient.password1;
  this.address=patient.address;
};

Patient.create = (newPatient,result) => {
  sql.query('insert into patient set ?',newPatient,(err,res) =>{
    if(err){
      console.log(err);
      result(err,null);
      return;
    }
    console.log("Created patient : ",{id:res.insertedId,...newPatient});
    return (null,{id:res.insertedId,...newPatient});
  })
};

module.exports = Patient;

```

Receptionist:

```

const sql = require('../models/db.js');

const Reception = function(reception){
  this.username = reception.username;
  this.emailid = reception.emailid;
  this.mobilenumber = reception.mobilenumber;
  this.password1=reception.password1;
  this.address=reception.address;
};

```

```

Reception.create = (newReception,result) => {
  sql.query('insert into reception set ?',newReception,(err,res) =>{
    if(err){
      console.log(err);
      result(err,null);
      return;
    }
    console.log("Created receptionist : ",{id:res.insertedId,...newReception});
    return (null,{id:res.insertedId,...newReception});
  })
};

module.exports = Reception;

```

Registration:

```

const sql = require('../models/db.js');

const Register = function(register){
  this.username = register.username;
  this.emailid = register.emailid;
  this.mobilenumber = register.mobilenumber;
  this.password1=register.password1;
  this.address=register.address;
};

Register.create = (newRegister,result) => {
  sql.query('insert into registers set
?',newRegister, (err,res) =>{
    if(err){
      console.log(err);
      result(err,null);
      return;
    }
  })
}

```

```

        console.log("Created Register : ",{id:res.insertedId,...newRegister});

        return (null,{id:res.insertedId,...newRegister});
    })
};

module.exports = Register;

```

Reports:

```

const sql = require('../models/db.js');

const Reports = function(reports){
    this.patientname = reports.patientname;
    this.mobilenumber = reports.mobilenumber;
    this.address=reports.address;
    this.status = reports.status;
};

Reports.create = (newReports,result) => {
    sql.query('insert into Reports set ?',newReports,(err,res) =>{
        if(err){
            console.log(err);
            result(err,null);
            return;
        }

        console.log("Created Report : ",{id:res.insertedId,...newReports});
        return (null,{id:res.insertedId,...newReports});
    })
};

module.exports = Reports;

```

Updation:

```

const sql = require('../models/db.js');

```

```

const Updation = function(updation){
  this.name = updation.name;
  this.feedback = updation.feedback;
};

Updation.create = (newupdation,result) => {
  sql.query('insert into updation set ?',newupdation,(err,res) =>{
    if(err){
      console.log(err);
      result(err,null);
      return;
    }
    console.log("Created updation : ",{id:res.insertedId,...newupdation});
    return (null,{id:res.insertedId,...newupdation});
  })
};

module.exports = Updation;

```

ROUTES:

Admin:

```

module.exports = app =>{
  const admins = require('../controllers/admin.controller.js');

  app.post ("/admins",admins.create);
}

```

Appointment:

```

module.exports = app =>{

```

```
const appointment =  
require('../controllers/appointment.controller.js');  
  
app.post ("/appointment", appointment.create);  
}
```

Attendance:

```
module.exports = app =>{  
  const attendenced = require('../controllers/attendanced.controller.js');  
  
  app.post ("/attendance", attendenced.create);  
}
```

Login:

```
module.exports = app =>{  
  const logins =  
  require('../controllers/login.controller.js');  
  
  app.post ("/login", logins.create);  
}
```

Nurse:

```
module.exports = app =>{  
  const nurses =  
  require('../controllers/nurse.controller.js');  
  
  app.post ("/nurses", nurses.create);  
}
```

```
}
```

Patient:

```
module.exports = app =>{  
  const patients = require('../controllers/patient.controller.js');  
  
  app.post ("/patient",patients.create);  
}
```

Receptionist:

```
module.exports = app =>{  
  const reception = require('../controllers/reception.controller.js');  
  app.post ("/reception",reception.create);  
}
```

Register:

```
module.exports = app =>{  
  const register = require('../controllers/register.controller.js');  
  
  //create a new register  
  app.post ("/register",register.create);  
}
```

Report:

```
module.exports = app =>{  
  const reports = require('../controllers/report.controller.js');  
  
  app.post ("/reports",reports.create);  
}
```

```
}
```

Updation:

```
module.exports = app =>{  
  const updation = require('../controllers/updation.controller.js');  
  
  app.post ("/updation",updation.create);  
}
```

CHAPTER-10 MAINTENANCE

Project maintenance:

The process of tracking and enabling project activities in accordance with the project plan is an essential, but often overlooked, factor in overall project success. After spending so much time in the planning phase, many project managers have a tendency to take a step back once the other members of the project team start their work, but experienced project managers know that project monitoring is every bit as important as project planning.

Enterprise projects, in particular, require a steady commitment to project maintenance, simply because they tend to have much longer durations than projects undertaken at smaller organizations. The longer a project runs, the more likely it becomes that a small deviation from the project plan will snowball into a serious issue as the project progresses. No matter what size projects you're managing today, an appreciation for project maintenance can only improve your chances of success. These strategies below can help you keep your eye on the prize even during the longest-running projects. For maintenance of the website:

- 1.The database has to be updated regularly according to new available information.
2. Redundant and false information must be removed from the database.
3. Newer versions of PHP and MYSQL can be used for upgradation of web sites
And to improve the overall performance of the system.

CHAPTER-11

FUTURE SCOPE AND FUTURE ENHANCEMENT

This project(hospital management)helps in knowing of information about hospital regarding doctors and patients.The admin of the hospital can easily know the information about ratio of the doctors and patients,doctors attendance,new joinings of hospital staff.It helps to know about the patients information too where admin can easily calculate no.of beds, machinery required for patients treatment and no.of patients recovered.

Project name:

MEDICINO(Hospital management)

CHAPTER-12

CONCLUSION

We have successfully implemented the site 'MEDICINO' with the help of various technologies and tools,we have been able to provide a site which will be live soon and running on the web. We have been successful in our attempt to take care of the needs of both the user as well as the administrator.Finally we hope that this will go a long way in popularizing.