

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1. Purpose

This application is for the automation of Doctor-Patient Interactions. Medical Information System is developed to support the daily operations, of a clinic/hospital. Traditionally this is done manually. This system will improve all the clinic operation starting from patient registration until doctor prescription and instructions. The important thing is it will become easier for the data record and retrieval. This system will be able to manage the complete operation inside a clinic/hospital. For example, patient appointments, doctor schedules and prescription instructions. This system is able to show the details for the medicine in the prescription. The target user for this system is staff of the clinic, doctor and also the patient. Prototyping approach is applied while developing this system. This will involve an iterative process to make this system is usable and easy to use by the user.

It maintains three types of users:

- Administrator
- Doctor
- Patient

The Software includes:

- Maintaining Patient details.
- Providing Prescription, Instructions and Diet advice.
- Providing and maintaining all kinds of tests for a patient.
- Schedule information.

1.2. Scope:

It can be used in any Hospital, Clinic, Dispensary or Pathology labs for maintaining patient details and their test results. This scope is convergent to the any clinic/hospital. The scope for the system will involve staff, doctor and patient of the clinic/hospital. The staff will register the patient. The doctor will diagnose the patients and give the medication while the patient will view the prescription and appointments as required.

1.3. Technologies to be used:

- Front-end: HTML5, CSS3, Bootstrap, JavaScript, jQuery
- Back-end: PHP, MySQL
- Other tools: Git, phpMyAdmin, VSCode

1.4. Overview

Project is related to Medical Information System.

The project maintains three types of users:-

- Administrator
- Doctor
- Patient

Main facilities available in this project are:-

- Maintaining records of indoor/outdoor patients.
- Maintaining patient's diagnosis details, advised tests to be done.
- Providing different facilities to a doctor for scheduling of patients.
- Providing different facilities to patient for managing appointments, prescriptions.
- Managing patients' vitals.
- Maintaining patient's prescription, medicine and appointment details.
- Maintaining backup of data as per user requirements.

- In this project collection of data is from different pathology labs.
- Results of tests, prescription, precautions and diet advice will be automatically updated in the database.
- Related test reports, patient details report, prescription and billing reports can be generated as per user requirements.

1.5. Any other information

The medical information system will improve clinic operation for both staff and the patient. For the patient, it will make it easy during registration process. If the patient is an existing patient, they can easily retrieve back the record of the patient. For the doctor they can view history record of patient. In case, if the patient allergy with the certain medicine, the doctor will give an alternative medicine for the patient. For the management, it will help them view operation of the clinic. The other thing is it will maintain the account for the clinic.

The definition and maintenance of the database is done through the phpMyAdmin server interface.

The version control is done through bitbucket/git.

CHAPTER 2

REQUIREMENT

CHAPTER 2

REQUIREMENT

2.1. Introduction

This application is for the automation of Doctor-Patient Interactions. Medical Information System is developed to support the daily operations, of a clinic/hospital. Traditionally this is done manually. This system will improve all the clinic operation starting from patient registration until doctor prescription and instructions. The important thing is it will become easier for the data record and retrieval. This system will be able to manage the complete operation inside a clinic/hospital. For example, patient appointments, doctor schedules and prescription instructions. This system is able to show the details for the medicine in the prescription. The target user for this system is staff of the clinic, doctor and also the patient. Prototyping approach is applied while developing this system. This will involve an iterative process to make this system is usable and easy to use by the user.

It maintains three types of users:

- Administrator
- Doctor
- Patient

The Software includes:

- Maintaining Patient details.
- Providing Prescription, Instructions and Diet advice.
- Providing and maintaining all kinds of tests for a patient.
- Schedule information.

2.2. Overall Description

It can be used in any Hospital, Clinic, Dispensary or Pathology labs for maintaining patient details and their test results. This scope is convergent to the any clinic/hospital. The scope for the system will involve staff, doctor and patient of the clinic/hospital. The staff will register the patient. The doctor will diagnose the patients and give the medication while the patient will view the prescription and appointments as required.

2.3. Specific Requirements

Front-end: HTML5, CSS3, Bootstrap, JavaScript, jQuery

Back-end: PHP, MySQL

Other tools: Git, phpMyAdmin, VSCode

CHAPTER 3

FEASIBILITY

CHAPTER 3

FEASIBILITY

3.1 Introduction

“FEASIBILITY STUDY” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources.

It focuses on these major questions:

1. What are the user’s demonstrable needs and how does a candidate system meet them?
2. What resources are available for given candidate system?
3. What are the likely impacts of the candidate system on the organization?
4. Whether it is worth to solve the problem?

During feasibility analysis for this project, following primary areas of interest are to be considered. Investigation and generating ideas about a new system does this.

Eight steps involved in the feasibility analysis are:

- Form a project team and appoint a project leader.
- Prepare system flowcharts.
- Enumerate potential proposed system.
- Define and identify characteristics of proposed system.
- Determine and evaluate performance and cost effective of each proposed system.
- Weight system performance and cost data.
- Select the best-proposed system.
- Prepare and report final project directive to management.

3.2 Technical feasibility

A study of resource availability that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not.

- Can the work for the project be done with current equipment existing software technology & available personal?
- Can the system be upgraded if developed?
- If new technology is needed then what can be developed?

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may include:

- Front-end and back-end selection

An important issue for the development of a project is the selection of suitable front-end and back-end. When we decided to develop the project, we went through an extensive study to determine the most suitable platform that suits the needs of the organization as well as helps in development of the project.

The aspects of our study included the following factors:

- Front-end selection:
 1. It must have a graphical user interface that assists employees that are not from IT background.
 2. Scalability and extensibility.
 3. Flexibility.
 4. Robustness.
 5. According to the organization requirement and the culture.
 6. Must provide excellent reporting features with good printing support.
 7. Platform independent.
 8. Easy to debug and maintain.
 9. Event driven programming facility.
 10. Front end must support some popular back end like MySQL.

According to the above stated features we selected HTML5, CSS3(Bootstrap), JavaScript(jQuery) as the front-end for developing our project.

- Back-end Selection:
 1. Multiple user support.
 2. Efficient data handling.
 3. Provide inherent features for security.
 4. Efficient data retrieval and maintenance.
 5. Stored procedures.
 6. Popularity.
 7. Operating System compatible.
 8. Easy to install.
 9. Various drivers must be available.
 10. Easy to implant with the Front-end.

According to above stated features we selected Apache (PHP) and phpMyAdmin (MySQL) as the backend. The technical feasibility is frequently the most difficult area encountered at this stage. It is essential that the process of analysis and definition be conducted in parallel with an assessment to technical feasibility. It centers on the existing computer system (scalability, security etc.) and to what extent it can support the proposed system.

3.3 Economic feasibility

Economic justification is generally the “Bottom Line” consideration for most systems. Economic justification includes a broad range of concerns that includes cost benefit analysis. In this we weight the cost and the benefits associated with the candidate system and if it suits the basic purpose of the organization i.e. profit making, the project is making to the analysis and design phase.

The financial and the economic questions during the preliminary investigation are verified to estimate the following:

- The cost to conduct a full system investigation.
- The cost of hardware and software for the class of application being considered.
- The benefits in the form of reduced cost.

- The proposed system will give the minute information, as a result the performance is improved which in turn may be expected to provide increased profits.
- This feasibility checks whether the system can be developed with the available funds. The Medical Information System does not require enormous amount of money to be developed.
- This can be done economically if planned judiciously, so it is economically feasible. The cost of project depends upon the number of man-hours required.

3.4 Operational Feasibility

It is mainly related to human organizations and political aspects. The points to be considered are:

- What changes will be brought with the system?
- What organization structures are disturbed?
- What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?
- The system is operationally feasible as it very easy for the end users to operate it. It only needs basic information about the Internet.

3.5 Schedule feasibility

Time evaluation is the most important consideration in the development of project. The time schedule required for the developed of this project is very important since more development time effect machine time, cost and cause delay in the development of other systems. A reliable Medical Information System can be developed in the considerable amount of time.

CHAPTER 4

SPECIFICATIONS

CHAPTER 4

SPECIFICATIONS

4.1. Introduction

Web application programming presents challenges that do not typically arise when programming traditional client-based applications. Among the challenges are:

- Implementing a rich Web user interface - It can be difficult and tedious to design and implement a user interface using basic HTML facilities, especially if the page has a complex layout, a large amount of dynamic content, and full-featured user-interactive objects. This is done by designing reusable HTML components, with client-side scripting and styling.
- Separation of client and server - In a Web application, the client (browser) and server are different programs often running on different computers (and even on different operating systems). Consequently, the two halves of the application share very little information; they can communicate, but typically exchange only small chunks of simple information. AJAX is used to make this process efficient
- Stateless execution - When a Web server receives a request for a page, it finds the page, processes it, sends it to the browser, and then discards all page information. If the user requests the same page again, the server repeats the entire sequence, reprocessing the page from scratch. Put another way, a server has no memory of pages that it has processed—page are stateless. Therefore, if an application needs to maintain information about a page, its stateless nature can become a problem.
- Unknown client capabilities - In many cases, Web applications are accessible to many users using different browsers. Browsers have different capabilities, making it difficult to create an application that will run equally well on all of them.
- Complications with data access - Reading from and writing to a data source in traditional Web applications can be complicated and resource-intensive.

- Complications with scalability - In many cases Web applications designed with existing methods fail to meet scalability goals due to the lack of compatibility between the various components of the application. This is often a common failure point for applications under a heavy growth cycle.

Meeting these challenges for Web applications can require substantial time and effort. Web Forms address these challenges in the following ways:

- Intuitive, consistent object model - The page framework presents an object model that enables you to think of your forms as a unit, not as separate client and server pieces. In this model, you can program the page in a more intuitive way than in traditional Web applications, including the ability to set properties for page elements and respond to events.
- Event-driven programming model - Web Forms bring to Web applications the familiar model of writing event handlers for events that occur on either the client or server.
- Intuitive state management - The page framework automatically handles the task of maintaining the state of your page and its controls, and it provides you with explicit ways to maintain the state of application-specific information. This is accomplished without heavy use of server resources and can be implemented with or without sending cookies to the browser.
- Browser-independent applications - The page framework enables you to create all application logic on the server, eliminating the need to explicitly code for differences in browsers. However, it still enables you to take advantage of browser-specific features by writing client-side code to provide improved performance and a richer client experience.

4.2. UI/UX – Front-end

User Interface (UI) Design has been designed with focus on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

Users can easily become familiar with interface elements acting in a certain way, as consistent and predictable choices and layouts are used. Doing so will help with task completion, efficiency, and satisfaction.

Interface elements include but are not limited to:

- Input Controls: buttons, text fields, checkboxes, radio buttons, dropdown lists, list boxes, toggles, date field
- Navigational Components: breadcrumb, slider, search field, pagination, slider, tags, icons
- Informational Components: tooltips, icons, progress bar, notifications, message boxes, modal windows
- Containers: canvas, blocks, strips

There are times when multiple elements are appropriate for displaying content, when this happens trade-offs are made for better experience. For example, sometimes elements that can help save space, put more of a burden on the user mentally by forcing them to guess what is within the dropdown or what the element might be.

- Everything stems from knowing our users, including understanding their goals, skills, preferences, and tendencies.
- Keeping the interface simple. The interface is almost invisible to the user. Unnecessary elements are avoided and clear language is used on labels and in messaging.
- Consistency and common UI elements. By using common elements in our UI, users feel more comfortable and are able to get things done more quickly. It is also important to create patterns in language, layout and design throughout the site to help facilitate efficiency. Once a user learns

how to do something, they should be able to transfer that skill to other parts of the site.

- Purposeful page layout. Considering the spatial relationships between items on the page and structuring the page based on importance. Careful placement of items can help draw attention to the most important pieces of information and can aid scanning and readability.
- Strategically using colour and texture. Attention is directed towards or redirected away from items using colour, light, contrast, and texture to our advantage.
- Use of typography to create hierarchy and clarity. Careful use of typefaces. Different sizes, fonts, and arrangement of the text to help increase scanability, legibility and readability.
- Making sure that the system communicates what's happening. Always inform our users of location, actions, changes in state, or errors. The use of various UI elements to communicate status reducing frustration for our user.
- Think about the defaults. By carefully thinking about and anticipating the goals people bring to your site, carefully created defaults that reduce the burden on the user. This becomes particularly important when it comes to form design where we have an opportunity to have some fields pre-chosen or filled out.

4.2.1 Pages

The interface is divided into various pages. This helps in proper separation of the task at hand. The different pages are used for different layouts and goals.

The different pages are:

4.2.1.1 Home/Index

Home page is crucially important for the website success and efficiency, having multiple functions: it's a card of invitation, a starting point of the journey around the website, a storage of the vital

links and data, and a strategic asset for marketing goals. In the majority of cases, it is often the first visual and emotional touch to the website.

Thus, the home page or landing page of the project is used for login and display of most popular tasks and entities of the platform. The most popular doctors, medicines and tests are displayed here.

4.2.1.2 Dashboard

This page is used for registered users to allow them to navigate the application. This page contains all the modules that the user can access based on his privileges. The doctor can access his patients, schedules and other necessary information. Similarly, the patient can access his doctors, appointments, prescriptions and other tasks. Apart from these, administrator also has his own dashboard with modules that help in activating users and creating new user accounts explicitly. Also, the dashboard contains option for accessing and updating user profiles.

4.2.1.3 Expanded Dashboard (My Page)

When the dashboard is expanded using the 'My' button on the navigation menu then the page goes into expanded view. Module expands from the sidebar to full page view for changing the focus to the immediate task at hand and removing the extra information.

4.2.2 Components

The pages are divided into components that are used on different pages in different layouts and forms to create personalised views for the current task at hand. An example of this is the navigation menu which is present on different dashboard and displays modules of the user based on his account type. The different components designed are:

- z-strip
- z-block

- z-card
- z-canvas
- z-impcard
- navbar

4.2.2.1 Navigation Menu

Usability is an essential goal of any website, and usable navigation is something every website needs. It determines where users are led and how they interact with the website. Without usable navigation, content becomes all but useless. Menus need to be simple enough for the user to understand, but should also contain the elements necessary to guide the user through the website — with some creativity and good design thrown in.

With the above points in mind the navigation menu is designed for the index, dashboard and expanded dashboard pages and help in navigation between various modules. The navigation bar indicates the active module as well as the available modules to the user. The navigation bar logo allows for expansion of the modules by removing the left strip and expanding the current module in focus.

4.2.2.2 Canvas

This serves as the container for all the information displayed on the webpage. It separates the content from the visual parts of the page in order to drive focus on the current scope of the task. It is the area where various components are laid out.

The canvas of the page is main area where all of the content of the website resides. It needs to be in contrast with the background in order to focus on the important aspect of the application. This canvas is divided into strips which in turn is divided into blocks that are further comprised of cards.

4.2.2.3 Forms

As we can find on Wikipedia, “Form refers to the shape, visual appearance, or configuration of an object.” And here I believe that this is also the perfect explanation... forms in web design are a way to gather information, but they must also take on a “form” or an appealing visual appearance in order to be effective.

4.2.2.4 Tables

Application user interfaces are defined using layouts, and table layouts are incredibly handy for displaying view data or controls in rows and columns. Using table layouts where they are appropriate can make many screen designs simpler and faster. Tables are the simplest method for creating layouts in HTML. Generally, this involves the process of putting the content into rows and columns. The tables in this application are rendered on the basis of a back-end renderer along with consistent styling that allows for code reuse and consistent design throughout the application.

4.2.2.5 Modules

In software engineering, modularity refers to the extent to which a software/Web application may be divided into smaller modules. Software modularity indicates that the number of application modules are capable of serving a specified business domain. Modularity makes complexity manageable, and enables parallel work and therefore this application is divided into modules. For consistency among different modules different components are designed once and used everywhere. When designing the modules brighter colours are used for calling attention to the important sections such as the action buttons of the specific modules.

4.2.2.6 Miscellaneous (ajax loader)

In an ideal world, pre-loaders should not exist. But in an ideal scenario, you want to inform your visitors that the web page is loading the resources it needs to show the web page in all its glory to the visitor. Since the website behind this loading animation is a brewery, this loader gif makes all the sense in the world. It depicts the production process of the asynchronously loading elements which will not only reduce the frustration of the users that have to wait for the page to load, but it will actually spark their interest in a creative way.

4.3. Server-side Render

Server-side rendering is the most common method for displaying information onto the screen. It works by converting script files in the server into usable information for the browser. Whenever you visit a website, your browser makes a request to the server that contains the contents of the website. The different components of the application are rendered using different php scripts in order to maintain consistency and reduce redundancy.

4.3.1 Canvas Painter

This server side php script facilitates rendering the complete canvas by calling the different component scripts on the backend and collating the end result into an html-based response that can be used to display components on the page. This is the most basic script used on all pages for laying out the modules, sub-navigation menus and other elements.

4.3.2 Form template

This server-side php script facilitates rendering any forms that may be used in the different modules throughout the application by generating automated html by using the variables in the script initialised by the parent script. Based on the values of the parameters form is generated containing various input fields, with both client-side and server-side validation properties, asynchronous submit properties and dynamic values as required.

4.3.3 Table template

This server-side php script facilitates rendering any tables that may be used in the different modules throughout the application by generating automated html by using the variables in the script initialised by the parent script. Based on the values of the parameters table is generated containing rows and columns, from the database as required.

4.3.4 Modules

It is very important for a complex application to be divided into smaller parts in order to achieve better results. It allows for efficient development processes with different people working on different aspects of the project developing the project parallelly from different use-case standpoints. It makes the software adhere to a larger feature set in a very short amount of time. It also allows for better separation of concerns of different types of users of the application, as the complex procedure is spread out into simpler and smaller tasks that can be managed easily at once. Thus, this project requires the separation of different concerns of the users into different modular components developed independently. It allows for better testing of the

product by creating unit and system tests. Keeping these ideas in mind this project is divided into modules, based on first the type of the user and secondly the goal he wants to achieve.

4.3.4.1 Admin modules

These are the modules that are accessible by the administrator. On the server these modules are present in the php-front/a directory, where 'a' denotes that the user type of the user is administrator and that they are allowed to access these modules, and can be accessed by making an AJAX request to php-back/getmodule.php with POST parameters containing the module requested. The getmodule script checks the session whether the user is logged in and returns the module html if the logged in user has this module in the respective directory based on his usertype.

The various admin modules are:

- approve.php – This module is used for approving the various users that have registered on the website.
- create.php – This module is used for creating new users explicitly, this feature can be used for creating pre-activated user. Also, this is the only method to create new users with the usertype 'a' or administrator.

4.3.4.2 Doctor modules

These are the modules that are accessible by the doctor. On the server these modules are present in the php-front/d directory, where 'd' denotes that the user type of the user is doctor and that they are

allowed to access these modules, and can be accessed by making an AJAX request to `php-back/getmodule.php` with POST parameters containing the module requested. The `getmodule` script checks the session whether the user is logged in and returns the module html if the logged in user has this module in the respective directory based on his usertype.

The various doctor modules are:

- `instruction.php` – This module is used for viewing all the prescriptions that this doctor has registered on the website regardless of the patient.
- `patient.php` – This module is used for viewing all the patients that this doctor has been involved with. This module allows managing the various patients that have ever had an appointment with the logged-in doctor.
- `profile.php` – This module is used for viewing and editing the profile of the logged-in doctor.
- `record.php` – This module is used for viewing all the appointments that this doctor has registered on the website for all his patients.
- `time.php` – This module is used for viewing all the schedules that this doctor has registered on the website that will be used by the patients to book appointments.

4.3.4.3 Patient modules

These are the modules that are accessible by the patient. On the server these modules are present in the php-front/p directory, where 'p' denotes that the user type of the user is patient and that they are allowed to access these modules, and can be accessed by making an AJAX request to php-back/getmodule.php with POST parameters containing the module requested. The getmodule script checks the session whether the user is logged in and returns the module html if the logged in user has this module in the respective directory based on his usertype.

The various patient modules are:

- instruction.php – This module is used for viewing all the prescriptions that this patient has registered on the website for all of his doctors.
- doctor.php – This module is used for viewing all the doctors that this patient has been involved with. This module allows managing the various doctors that have ever had an appointment with the logged-in patient.
- profile.php – This module is used for viewing and editing the profile of the logged-in patient.
- record.php – This module is used for viewing all the appointments that this patient has registered on the website for all his doctors.

4.4. Application Logic

The application logic is separated from other html rendering scripts for separation of concerns. These scripts are used to implement the business logic of the application. It is the part where we query, update and insert data to/from the database. These scripts also help in server-side validation scripts for sanitization of input data. This helps addressing concerns of code injection through input fields. This portion of the project also contains code for session handling, database connections, user management and functions for encryption, input validation and email notifications.

4.4.1 Database Connection

The application logic requires connection to a SQL based database at various times during execution of the application functions. To avoid redundancy and increase maintainability of code the connection to database is done using a common php script that contains the name of the database the username and password for connection as well as other parameters such as the address and port numbers. This allows for smooth transition between databases in case there is a change of environment at a later stage of development, or even at the time of deployment to production.

4.4.2 Dashboard Render

The dashboard renderer uses preset variables by the parent script, the session and POST parameters to determine the dashboard requested by the user. After determining the type of the user in question it fetches the specific modules

from the php-server directory and returns the first module along with the name and identifiers of the eligible modules as the response.

4.4.3 Module Redirector

This application script checks the POST and session parameters to determine the requested parameter and makes sure the user is logged in and then according to the POST parameters determines the module requested. After determining the type of user, module as well as the module parameters the redirector fetches the module from the respective user directory, cooks up the html and sends it as the response for the asynchronous request. The client uses javascript to replace the current module visible by the new response, and replaces the active module on the navigation menu, accordingly.

4.4.4 Functions

Any application requires logic functions to execute common tasks throughout the application. These functions are used throughout the application to implement the business logic. Rewriting this code everywhere as required would create a very complex codebase which is very hard to maintain. To avoid code redundancy and improve maintainability the concept of modularity is used to create a set of commonly used functions that are used by different scripts throughout the application. These functions achieve tasks such as form validation, database handling (querying, updating and inserting data to the database), data sanitization (input validation, password hashing) and other logic required for running the other scripts.

4.4.5 Form Validation

These functions allow for validation of various inputs send by the client. It helps in implementing security features such as prevention of code injection through input fields, preventing SQL injections and encrypting user data when required. It also helps in determining the errors in the inputs provided by the user and notifying them.

4.4.6 Submit Form

These functions help in taking all the inputs received from the user through the asynchronous POST request and use them to update the database accordingly. The use of functions from the functions file is required and data or input validation is done is using validation rules defined using php variables in this file. Specific rules are applied for specific forms and inputs using parameters like max-length, min-length, exact-length and alphanum etc. This helps in restricting the input both on the basis of what characters are allowed and length of specific inputs. It is done because storage resources are not infinite and to properly manage them an upper limit is required. In some cases a lower limit may be required for security and syntactic purposes.

4.5. Database Structure

A good database design is important in ensuring consistent data, elimination of data redundancy, efficient execution of queries and high-performance application. Taking the time to design a database saves time and frustration during development, and a well-designed database ensures ease of access and retrieval of information.

Logical database design is the process of deciding how to arrange the attributes of the entities in a given business environment into database structures, such as the

tables of a relational database. The goal of logical database design is to create well-structured tables that properly reflect the application's business environment. Thus, logical database design is used to create database that is robust, consistent and simple to implement.

Keeping these points in mind every table has the following types of fields:

4.5.1 Basic

This is the portion of the database which is mostly invisible to the user and the program and is done on the database platform to arrange data in a manner that allows for better maintenance procedures. The inclusion of an auto increment field for serial number allows for a primary key which is independent of the user input values and thus helps in resolving conflicts in data. Also automatically calculated timestamps for creation and latest modification of the data values helps in managing the data better.

4.5.2 Profile

This is user specific data that only belongs to the specific entity in question. For a patient this might be his name, contact details; similarly for a doctor these may be his name, qualifications, specialization and for medicine this portion contains a short description along with its name and so on.

4.5.3 Properties

This is the portion where the relation between different entities is stored. A patient's properties may contain the doctors he is connected to, his prescriptions; similarly a doctor's properties may contain his patients, his schedule etc.

CHAPTER 5

DATA FLOW DIAGRAMS

CHAPTER 5

DATA FLOW DIAGRAMS

5.1 Doctor

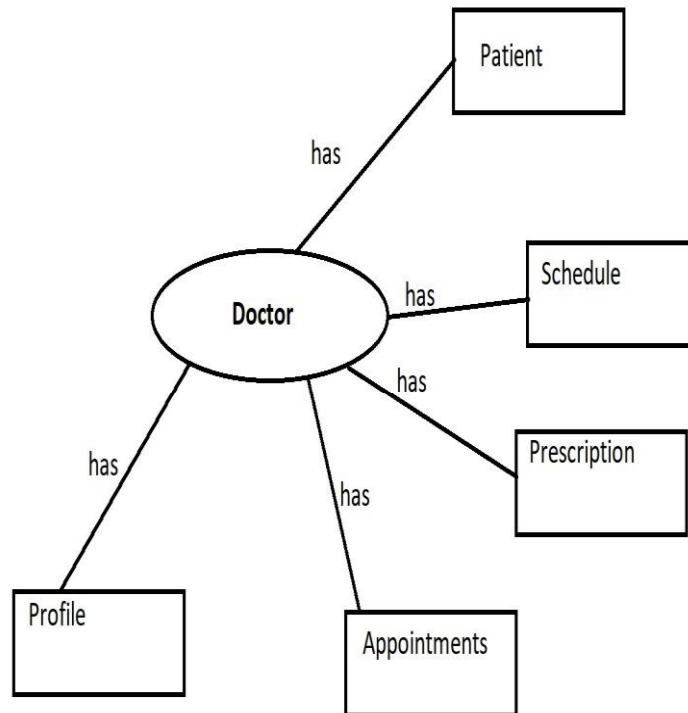


Fig 5.1 Doctor dataflow diagram

The various doctor modules are:

instruction.php – This module is used for viewing all the prescriptions that this doctor has registered on the website regardless of the patient.

patient.php – This module is used for viewing all the patients that this doctor has been involved with. This module allows managing the various patients that have ever had an appointment with the logged-in doctor.

profile.php – This module is used for viewing and editing the profile of the logged-in doctor.

record.php – This module is used for viewing all the appointments that this doctor has registered on the website for all his patients.

time.php – This module is used for viewing all the schedules that this doctor has registered on the website that will be used by the patients to book appointments.

5.2 Patient

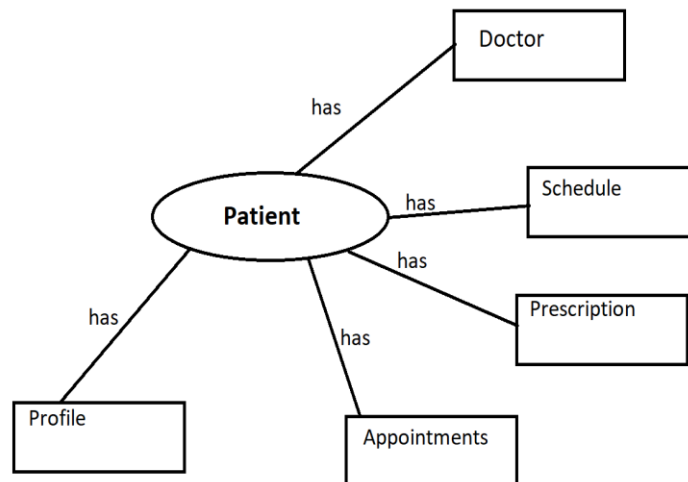


Fig 5.2 Patient dataflow diagram

The various patient modules are:

instruction.php – This module is used for viewing all the prescriptions that this patient has registered on the website for all of his doctors.

doctor.php – This module is used for viewing all the doctors that this patient has been involved with. This module allows managing the various doctors that have ever had an appointment with the logged-in patient.

profile.php – This module is used for viewing and editing the profile of the logged-in patient.

record.php – This module is used for viewing all the appointments that this patient has registered on the website for all his doctors.

5.3 Administrator

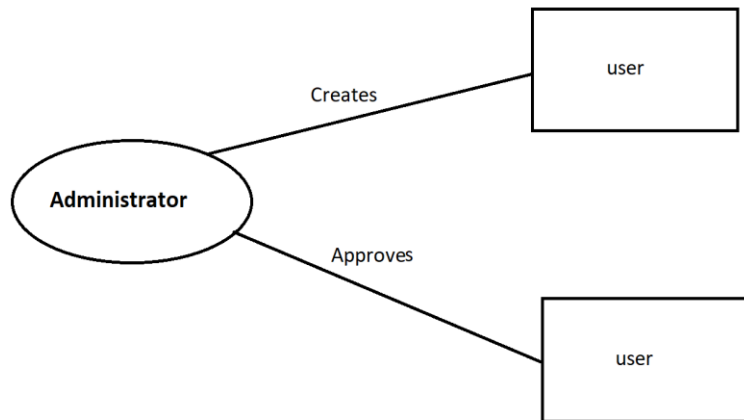


Fig 5.3 Admin dataflow diagram

The various admin modules are:

approve.php – This module is used for approving the various users that have registered on the website.

create.php – This module is used for creating new users explicitly, this feature can be used for creating pre-activated user. Also, this is the only method to create new users with the usertype 'a' or administrator.

CHAPTER 6

ENTITY RELATION DIAGRAMS

CHAPTER 6

ENTITY RELATION DIAGRAMS

6.1 Introduction

A good database design is important in ensuring consistent data, elimination of data redundancy, efficient execution of queries and high-performance application. Taking the time to design a database saves time and frustration during development, and a well-designed database ensures ease of access and retrieval of information.

Logical database design is the process of deciding how to arrange the attributes of the entities in a given business environment into database structures, such as the tables of a relational database. The goal of logical database design is to create well-structured tables that properly reflect the application's business environment. Thus, logical database design is used to create database that is robust, consistent and simple to implement.

Keeping these points in mind every table has the following types of fields:

1 Basic

This is the portion of the database which is mostly invisible to the user and the program and is done on the database platform to arrange data in a manner that allows for better maintenance procedures. The inclusion of an auto increment field for serial number allows for a primary key which is independent of the user input values and thus helps in resolving conflicts in data. Also automatically calculated timestamps for creation and latest modification of the data values helps in managing the data better.

2 Profile

This is user specific data that only belongs to the specific entity in question. For a patient this might be his name, contact details; similarly, for a doctor these may be his name, qualifications, specialization and for medicine this portion contains a short description along with its name and so on.

3 Properties

This is the portion where the relation between different entities is stored. A patient's properties may contain the doctors he is connected to, his prescriptions; similarly, a doctor's properties may contain his patients, his schedule etc.

6.2 Doctor

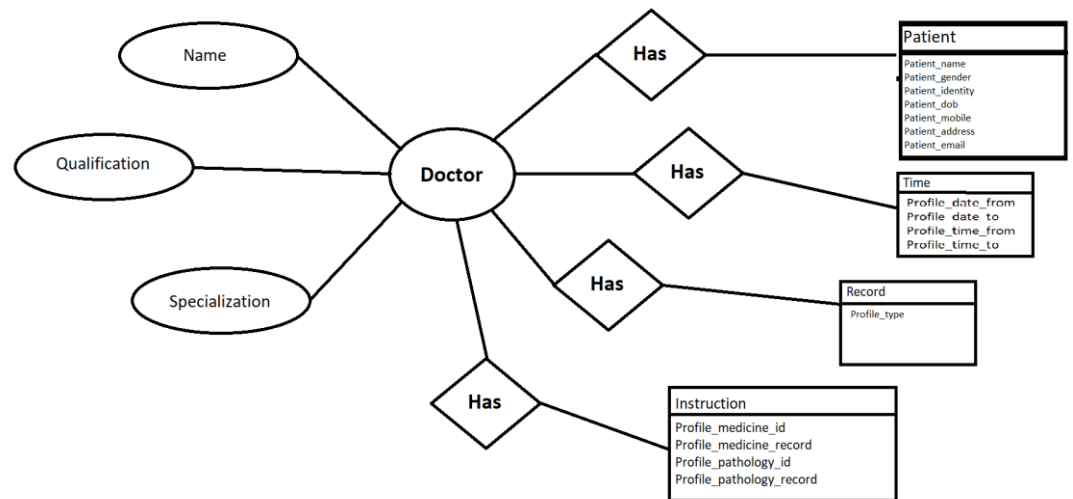


Fig 6.1 Database diagram for doctor

This is user specific data that only belongs to the specific entity in question. For a doctor these are his name, qualifications, specialization.

6.3 Patient

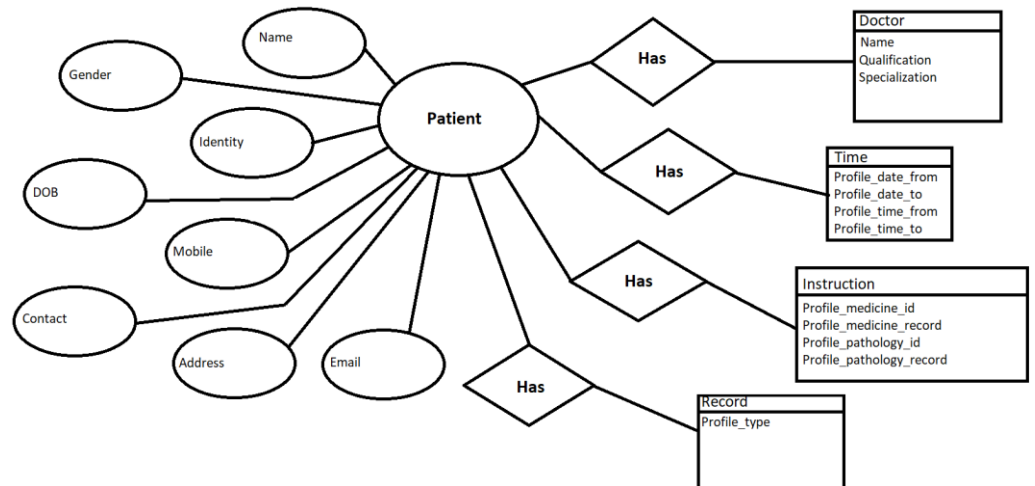


Fig 6.2 Database diagram for patient

This is user specific data that only belongs to the specific entity in question. For a patient these are his name, contact details.

6.4 Pathology

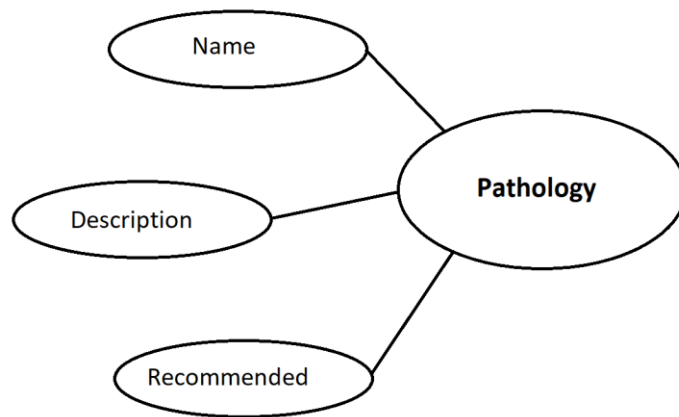


Fig 6.3 Database diagram for pathology

6.5 Record

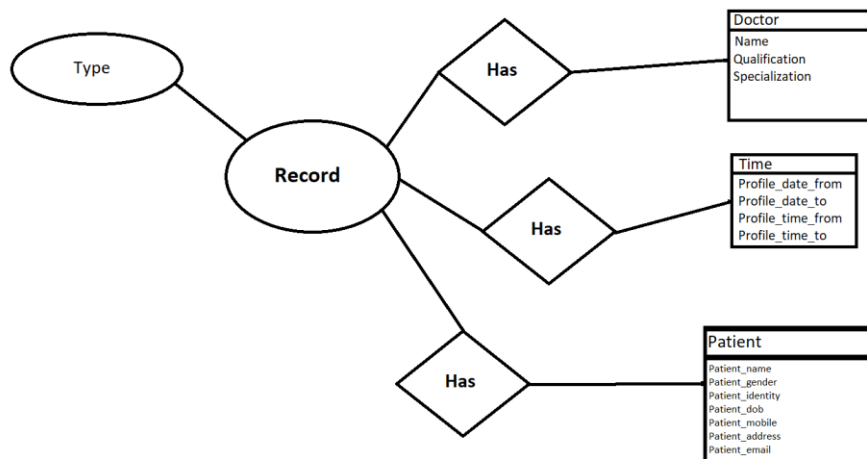


Fig 6.4 Database diagram for record

6.6 Medicine

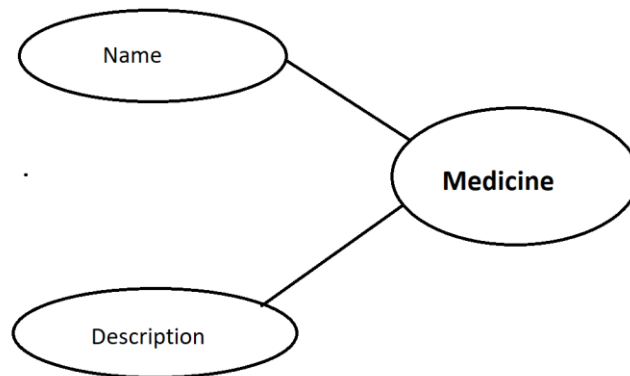


Fig 6.5 Database diagram for medicine

6.7 Instruction

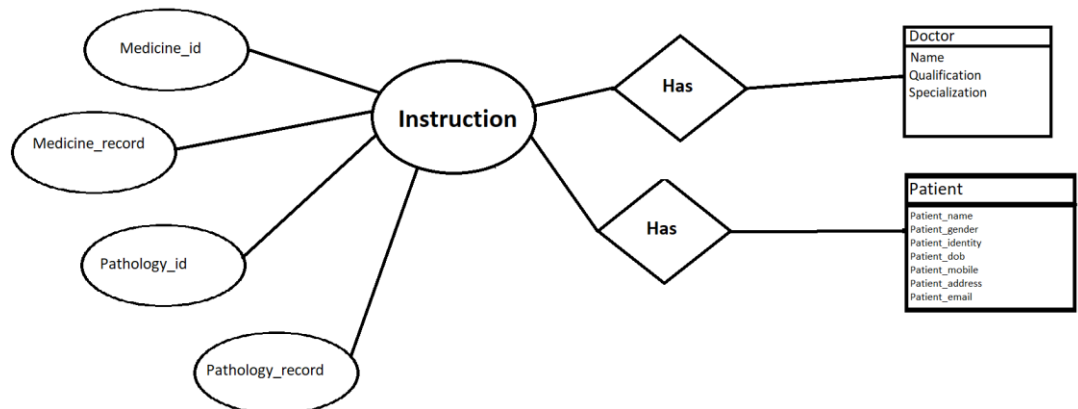


Fig 6.6 Database diagram for instruction

6.8 Time

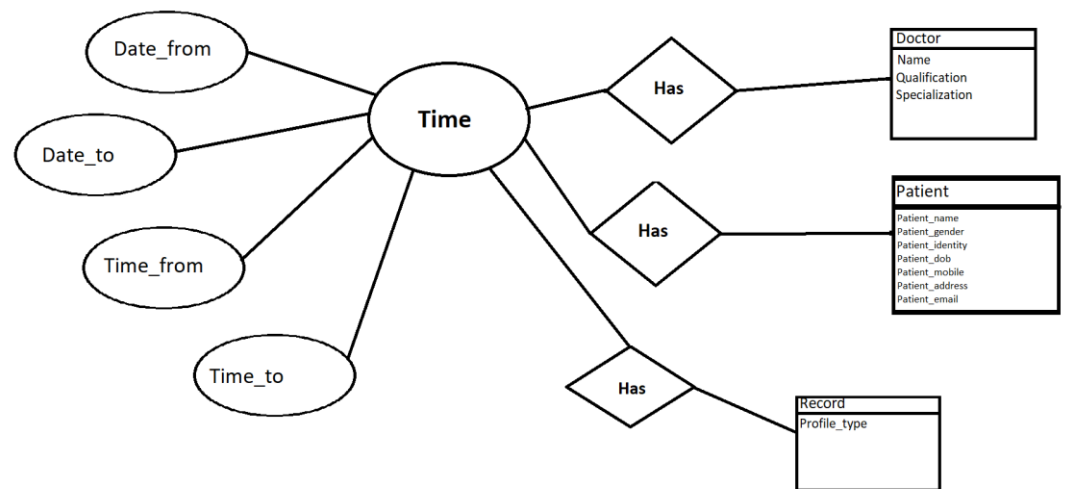


Fig 6.7 Database diagram for time

CHAPTER 7

DATABASE DESIGN DIAGRAMS

CHAPTER 7

DATABASE DESIGN DIAGRAMS

7.1 Introduction

A good database design is important in ensuring consistent data, elimination of data redundancy, efficient execution of queries and high-performance application. Taking the time to design a database saves time and frustration during development, and a well-designed database ensures ease of access and retrieval of information.

Logical database design is the process of deciding how to arrange the attributes of the entities in a given business environment into database structures, such as the tables of a relational database. The goal of logical database design is to create well-structured tables that properly reflect the application's business environment. Thus, logical database design is used to create database that is robust, consistent and simple to implement.

Keeping these points in mind every table has the following types of fields:

- **Basic**

This is the portion of the database which is mostly invisible to the user and the program and is done on the database platform to arrange data in a manner that allows for better maintenance procedures. The inclusion of an auto increment field for serial number allows for a primary key which is independent of the user input values and thus helps in resolving conflicts in data. Also automatically calculated timestamps for creation and latest modification of the data values helps in managing the data better.
- **Profile**

This is user specific data that only belongs to the specific entity in question. For a patient this might be his name, contact details; similarly, for a doctor these may be his name, qualifications, specialization and for medicine this portion contains a short description along with its name and so on.
- **Properties**

This is the portion where the relation between different entities is stored. A patient's properties may contain the doctors he is connected to, his prescriptions; similarly, a doctor's properties may contain his patients, his schedule etc.

7.2 Doctor

```

CREATE TABLE doctor (
  sno INT(11) AUTO_INCREMENT PRIMARY KEY,
  created TIMESTAMP,
  last_modified TIMESTAMP,
  id VARCHAR(20),
  profile_name VARCHAR(200),
  profile_qualification VARCHAR(200),
  profile_specialization VARCHAR(200),
  props_patient VARCHAR(2000),
  props_time VARCHAR(2000),
  props_record VARCHAR(2000),
  props_instruction VARCHAR(2000)
);

```

Doctor table has the following types of fields:

- Basic
 - sno – AUTOINCREMENT, Primary key, used for identifying individual entries and avoid any clashes occurring due to user input.
 - created – To store the timestamp when this entry was created.
 - last_modified – To store the timestamp of last edit.
 - id – This is unique identifier visible to the user. Username of the doctor is its id.
- Profile
 - name – Full name of the user.
 - qualification – Qualification of the user.
 - specialization – To store the timestamp of last edit.
- Properties
 - patient – Patients connected to this doctor.
 - time – Time-lapses connected to this doctor.
 - record – Appointments connected to this doctor.
 - instruction – Prescriptions connected to this doctor

7.3 Patient

```

patient
sno : int(11)
created : timestamp
last_modified : timestamp
id : varchar(20)
profile_name : varchar(200)
profile_gender : varchar(20)
profile_dob : varchar(20)
profile_identity : varchar(20)
profile_moblie : bigint(10)
profile_email : varchar(20)
profile_address : varchar(50)
profile_vitals : varchar(200)
props_doctor : varchar(2000)
props_time : varchar(2000)
props_record : varchar(2000)
props_instruction : varchar(2000)

```

Patient table has the following types of fields:

- Basic
 - sno – AUTOINCREMENT, Primary key, used for identifying individual entries and avoid any clashes occurring due to user input.
 - created – To store the timestamp when this entry was created.
 - last_modified – To store the timestamp of last edit.
 - id – This is unique identifier visible to the user. Username of the patient is its id.
- Profile
 - name – Full name of the user.
 - gender – Qualification of the user.
 - dob – To store the timestamp of last edit.
 - identity – To store identification document number of the patient.
 - mobile – To store contact information of the patient.
 - email – To store contact email of the patient.

- address – To store the contact address of the patient.
- vitals – To store the vitals of the patient.
- Properties
 - doctor – Doctors connected to this patient.
 - time – Time-lapses connected to this patient.
 - record – Appointments connected to this patient.
 - instruction – Prescriptions connected to this patient.

7.4 Instruction



Table instruction

```

sno : int(11)
created : timestamp
last_modified : timestamp
id : varchar(20)
profile_medicine_id : varchar(2000)
profile_medicine_record : varchar(200)
profile_pathology_id : varchar(2000)
profile_pathology_record : varchar(200)
profile_next : varchar(200)
props_doctor : varchar(200)
props_patient : varchar(200)
  
```

Instruction table has the following types of fields:

- Basic
 - sno – AUTOINCREMENT, Primary key, used for identifying individual entries and avoid any clashes occurring due to user input.
 - created – To store the timestamp when this entry was created.
 - last_modified – To store the timestamp of last edit.
 - id – This is unique identifier visible to the user. Unique identifier of the instruction its id.
- Profile
 - medicine_id – Medicines connected to this prescription.
 - medicine_record – Instructions related to the medicines on this prescription.
 - pathology_id – Pathology tests connected to this prescription.
 - pathology_record – Instructions related to the pathology on this prescription.
- Properties
 - doctor – Doctor connected to this instruction.
 - patient – Patient connected to this instruction.
 - next – Next appointment connected for this patient.

7.5 Record



CREATE TABLE record

```

sno : int(11)
created : timestamp
last_modified : timestamp
id : varchar(20)
profile_type : varchar(20)
props_doctor : varchar(2000)
props_patient : varchar(2000)
props_time : varchar(2000)
  
```

Record table has the following types of fields:

- Basic
 - sno – AUTOINCREMENT, Primary key, used for identifying individual entries and avoid any clashes occurring due to user input.
 - created – To store the timestamp when this entry was created.
 - last_modified – To store the timestamp of last edit.
 - id – This is unique identifier visible to the user. Unique identifier of the instruction its id.
- Profile
 - type – Type of record, medicine or pathology.
- Properties
 - doctor – Doctor connected to this record.
 - patient – Patient connected to this record.
 - time – Schedule connected for this record.

7.6 Time



time

```

sno : int(11)
created : timestamp
last_modified : timestamp
id : varchar(20)
profile_date_from : varchar(20)
profile_date_to : varchar(20)
profile_time_from : varchar(20)
profile_time_to : varchar(20)
profile_next : varchar(200)
props_doctor : varchar(2000)
props_patient : varchar(2000)
props_record : varchar(2000)
    
```

Time table has the following types of fields:

- Basic
 - sno – AUTOINCREMENT, Primary key, used for identifying individual entries and avoid any clashes occurring due to user input.
 - created – To store the timestamp when this entry was created.
 - last_modified – To store the timestamp of last edit.
 - id – This is unique identifier visible to the user. Unique identifier of the instruction its id.
- Profile
 - date_from – Start date of this schedule entry.
 - date_to – End date of this schedule entry.
 - time_from – Start time of this schedule entry.
 - time_to – End time of this schedule entry.
 - next – Next schedule entry connected to this schedule.
- Properties
 - doctor – Doctor connected to this schedule entry.
 - patient – Patient connected to this schedule entry.
 - record – Schedule connected for this schedule entry.

7.7 Medicine



medicine

🔑 sno : int(11)

📅 created : timestamp

📅 last_modified : timestamp

🔑 id : varchar(20)

profile_name : varchar(200)

profile_description : varchar(2000)

Medicine table has the following types of fields:

- Basic
 - sno – AUTOINCREMENT, Primary key, used for identifying individual entries and avoid any clashes occurring due to user input.
 - created – To store the timestamp when this entry was created.
 - last_modified – To store the timestamp of last edit.
 - id – This is unique identifier visible to the user. Unique identifier of the instruction its id.
- Profile
 - name – Name of the medicine.
 - description – Description of the medicine.

7.8 Pathology



pathology

```

sno : int(11)
created : timestamp
last_modified : timestamp
id : varchar(20)
profile_name : varchar(200)
profile_description : varchar(2000)
profile_recommended : varchar(2000)
    
```

Pathology table has the following types of fields:

- Basic
 - sno – AUTOINCREMENT, Primary key, used for identifying individual entries and avoid any clashes occurring due to user input.
 - created – To store the timestamp when this entry was created.
 - last_modified – To store the timestamp of last edit.

- id – This is unique identifier visible to the user. Unique identifier of the Pathology its id.
- Profile
 - name – Name of the pathology test.
 - description – Details about the pathology test.
 - recommended – Recommended limits of the pathology test.

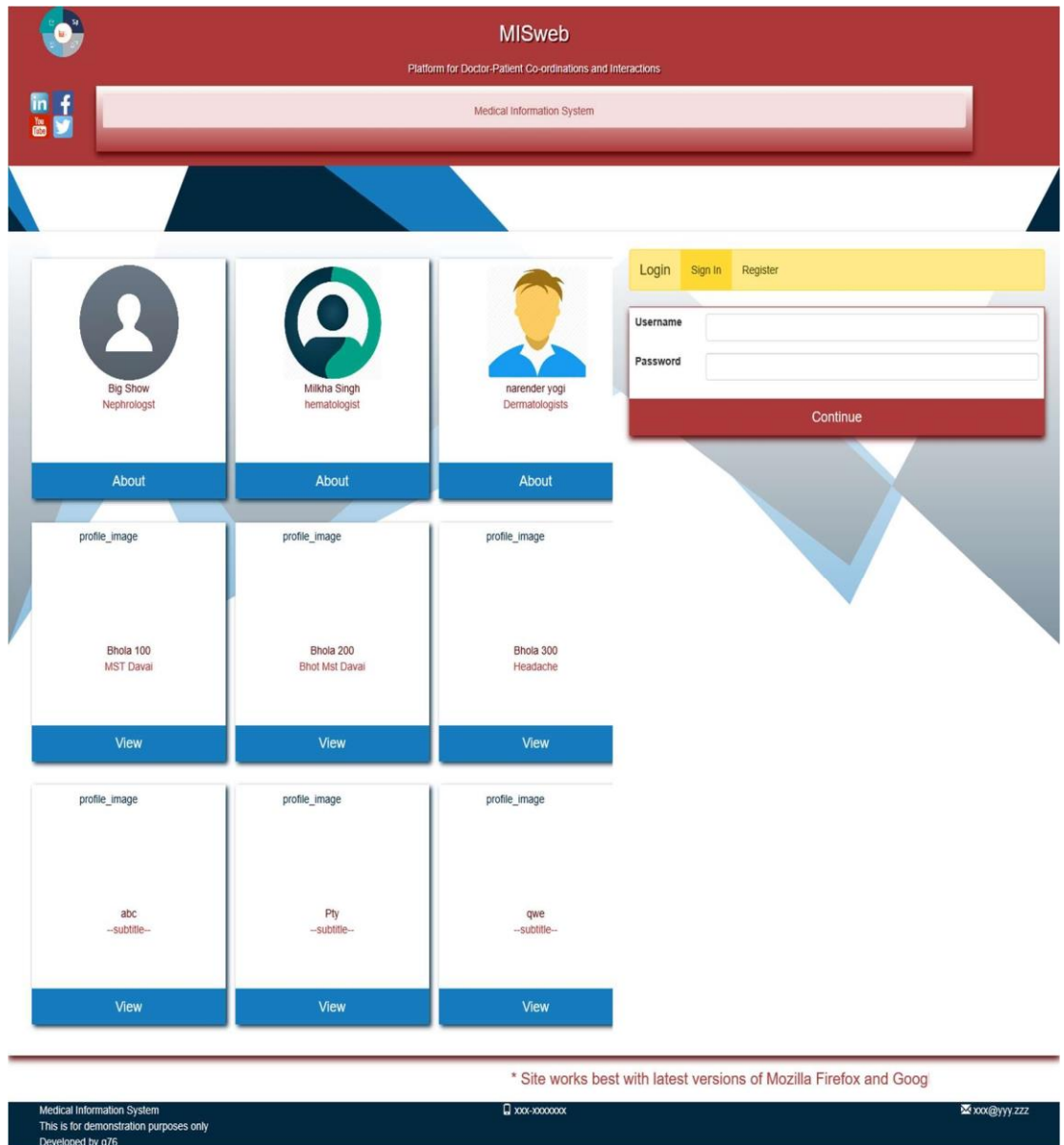
CHAPTER 8

SNAPSHOTS

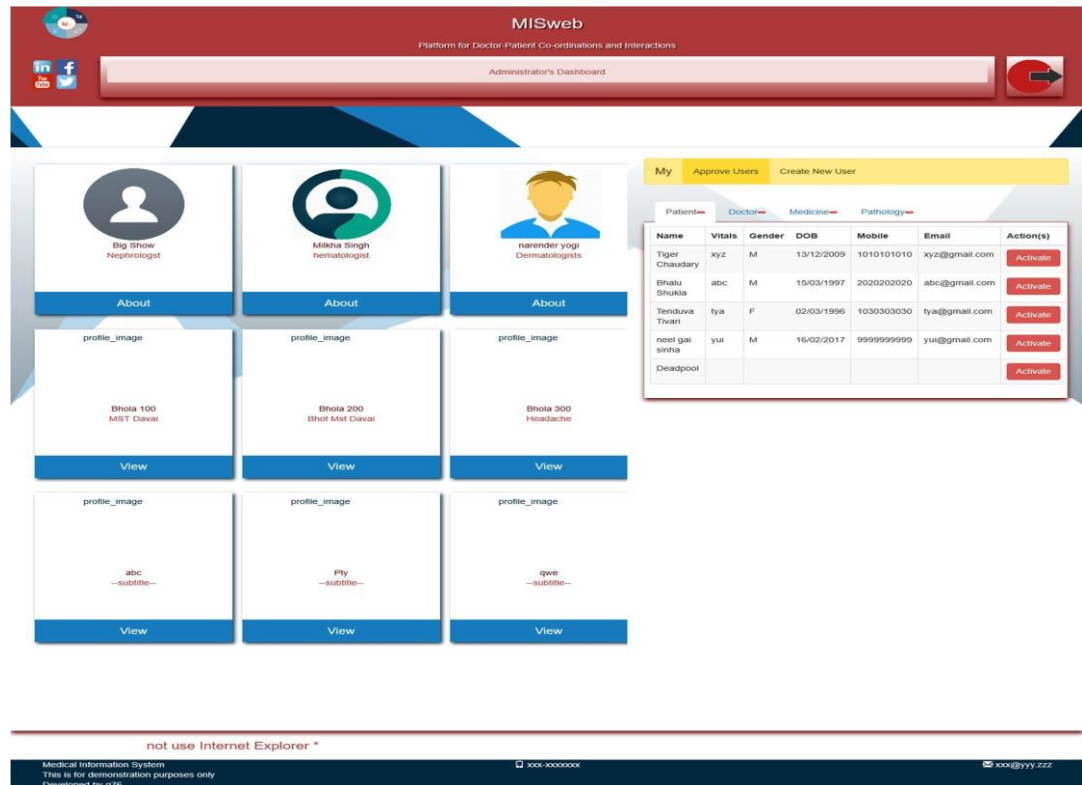
CHAPTER 8

SNAPSHOTS

8.1 Index Page



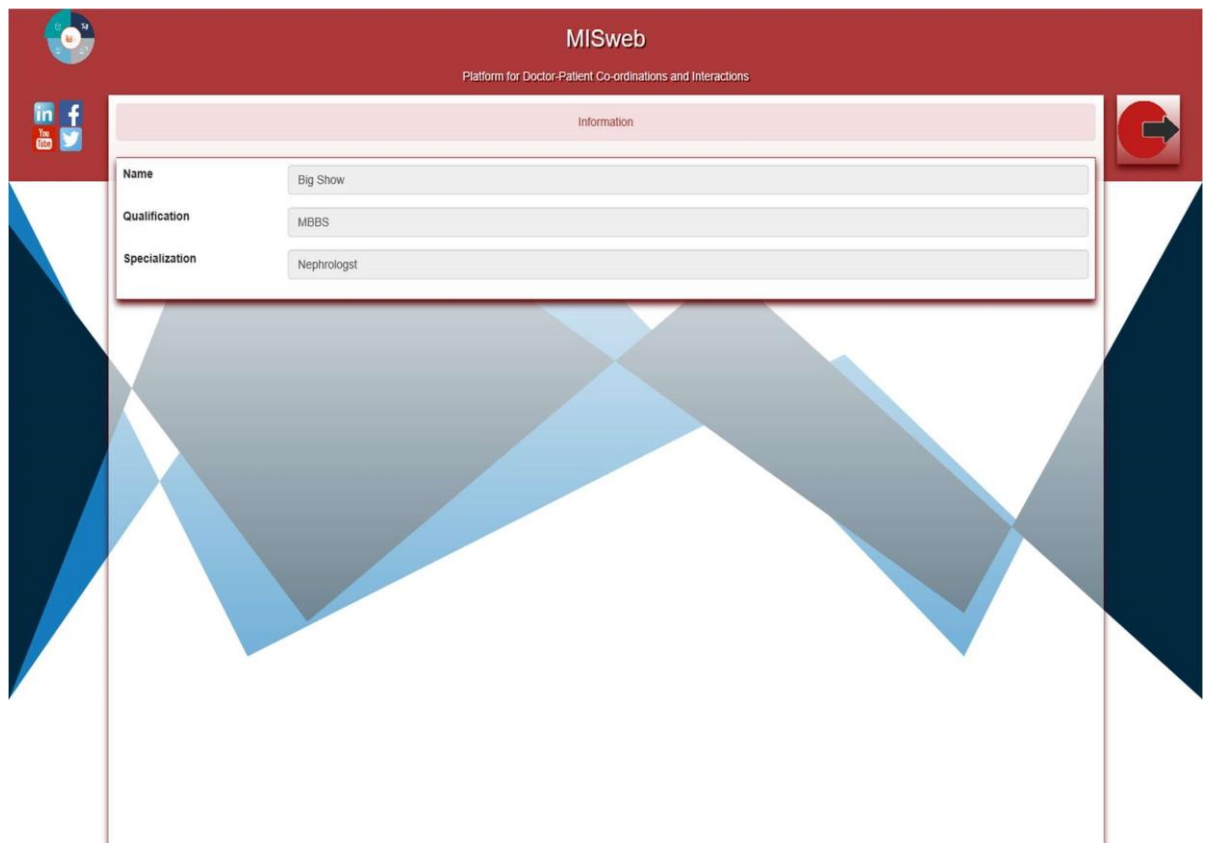
8.2 Dashboard – Administrator - Approve – Patients



The screenshot shows the MISweb Administrator Dashboard. The header includes the MISweb logo, social media icons, and a search bar. The main content area features three user profile cards for 'Big Show Nephrologist', 'Mikha Singh hematologist', and 'narendar yogi Dermatologists'. Each card has an 'About' section and a 'View' button. To the right, there is a table of patients with columns for Name, Vitals, Gender, DOB, Mobile, Email, and Action(s). The table lists several patients, including 'Tiger Chauhary', 'Bhuku Shukla', 'Tendruva Thani', 'neet gal sinha', and 'Deadpool'. At the bottom, there is a footer with a warning not to use Internet Explorer and a copyright notice.

Name	Vitals	Gender	DOB	Mobile	Email	Action(s)
Tiger Chauhary	xyz	M	13/12/2009	1010101010	xyz@gmail.com	Activate
Bhuku Shukla	abc	M	15/03/1997	2020202020	abc@gmail.com	Activate
Tendruva Thani	tya	F	02/03/1996	1030303030	tya@gmail.com	Activate
neet gal sinha	yui	M	16/02/2017	9999999999	yui@gmail.com	Activate
Deadpool						Activate

8.3 Information – Doctor



The screenshot shows the MISweb Information form for a Doctor. The form has a red header with the MISweb logo and social media icons. The main content area contains a form with three input fields: 'Name' (Big Show), 'Qualification' (MBBS), and 'Specialization' (Nephrologist). The form is titled 'Information' and has a 'Go' button on the right.

8.4 Dashboard – Administrator - Approve – Doctors

MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

Administrator's Dashboard

My Approve Users Create New User

Patient Doctor Medicine Pathology

Name	Qualification	Specialization	Action(s)
Osama kashyap	MBBS	cardiologists	Activate
narender yogi	MBBS	Dermatologists	Activate
Mikha Singh	MBBS	hematologist	Activate
Big Show	MBBS	Nephrologist	Activate

8.5 Dashboard – Administrator - Approve – Medicine

MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

Administrator's Dashboard

My Approve Users Create New User

Patient Doctor Medicine Pathology

Name	Description
Bhola 100	MST Davai
Bhola 200	Bhot Mst Davai
Bhola 300	Headache
Bhola 400	jabardast

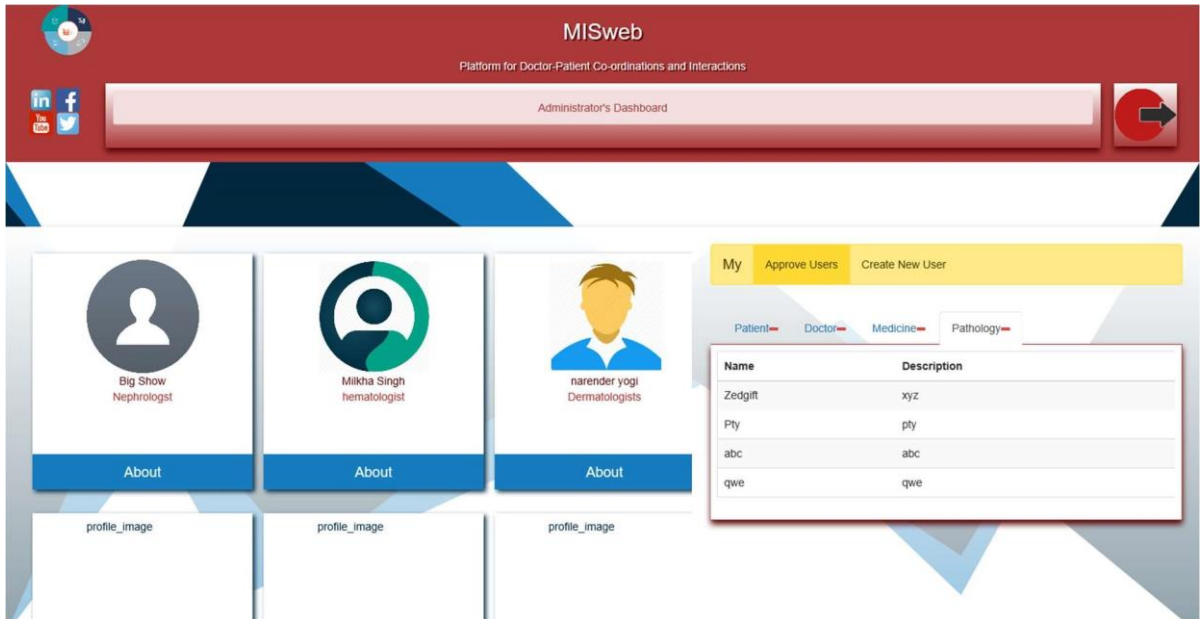
Please do not use Internet Explorer *

Medical Information System
This is for demonstration purposes only
Developed by g76

xxx-xxxxxx

xxx@yyy.zzz

8.6 Dashboard – Administrator - Approve – Pathology



MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

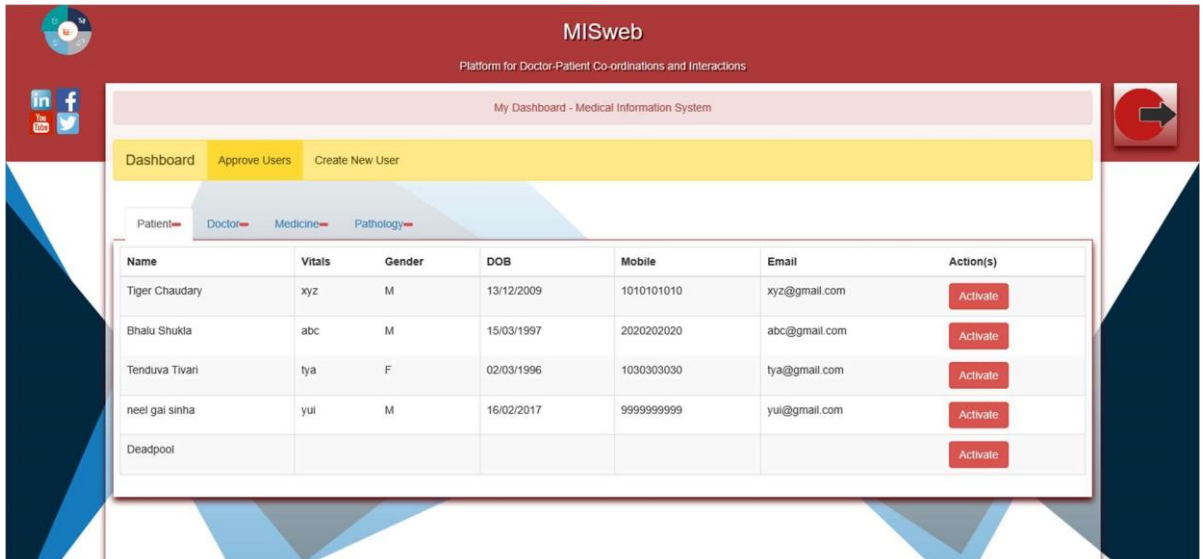
Administrator's Dashboard

My Approve Users Create New User

Patient Doctor Medicine Pathology

Name	Description
Zedgift	xyz
Pty	pty
abc	abc
qwe	qwe

8.7 Dashboard Expanded– Administrator - Approve – Patients



MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

My Dashboard - Medical Information System

Dashboard Approve Users Create New User

Patient Doctor Medicine Pathology

Name	Vitals	Gender	DOB	Mobile	Email	Action(s)
Tiger Chaudary	xyz	M	13/12/2009	1010101010	xyz@gmail.com	Activate
Bhalu Shukla	abc	M	15/03/1997	2020202020	abc@gmail.com	Activate
Tenduva Tivari	tya	F	02/03/1996	1030303030	tya@gmail.com	Activate
neel gai sinha	yui	M	16/02/2017	9999999999	yui@gmail.com	Activate
Deadpool						Activate

8.8 Dashboard Expanded– Administrator - Approve – Doctor

MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

My Dashboard - Medical Information System

Dashboard Approve Users Create New User

Patient Doctor Medicine Pathology

Name	Qualification	Specialization	Action(s)
Osama kashyap	MBBS	cardiologists	Activate
narender yogi	MBBS	Dermatologists	Activate
Milkha Singh	MBBS	hematologist	Activate
Big Show	MBBS	Nephrologst	Activate

8.9 Dashboard Expanded– Administrator - Approve – Medicine

MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

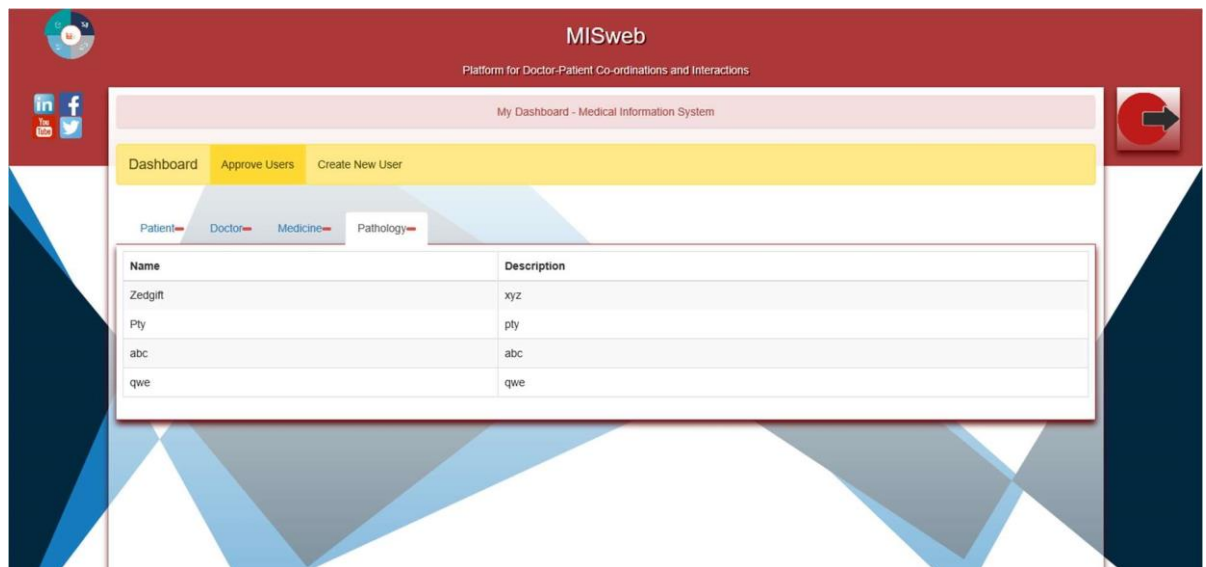
My Dashboard - Medical Information System

Dashboard Approve Users Create New User

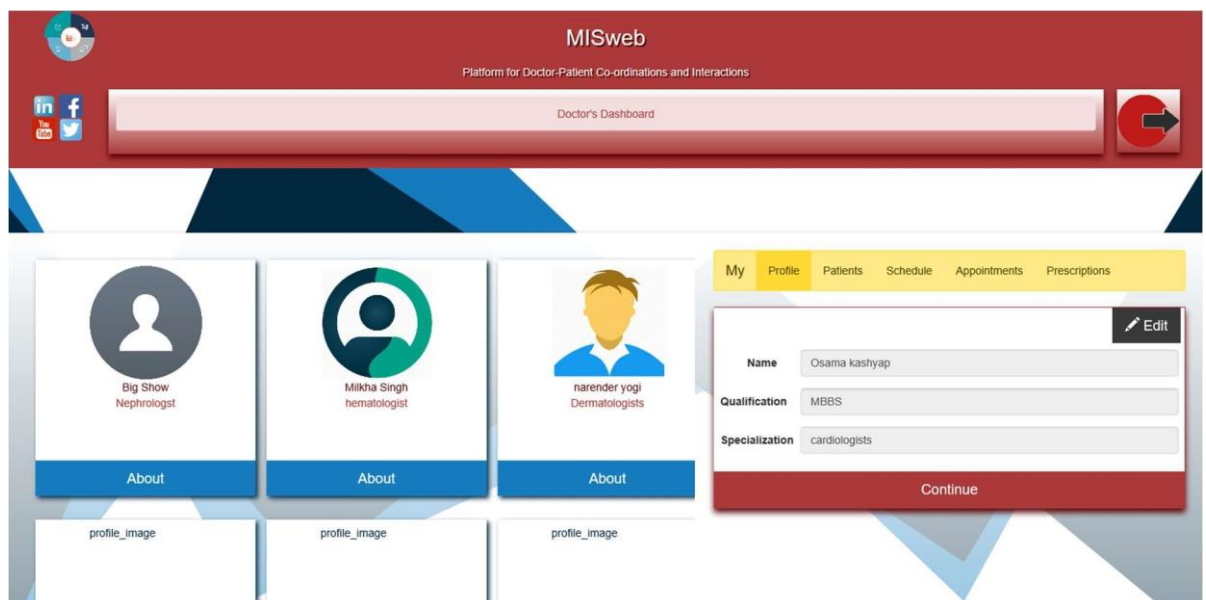
Patient Doctor Medicine Pathology

Name	Description
Bhola 100	MST Davai
Bhola 200	Bhot Mst Davai
Bhola 300	Headache
Bhola 400	jabardast

8.10 Dashboard Expanded– Administrator - Approve – Pathology



8.11 Dashboard–Doctor – Profile



8.12 Dashboard–Doctor – Patients

MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

Doctor's Dashboard

My Profile Patients Schedule Appointments Prescriptions

Name	Vitals	Gender	DOB	Mobile	Email	Action(s)
Tiger Chaudary	xyz	M	13/12/2009	1010101010	xyz@gmail.com	Presc.
Bhalu Shukla	abc	M	15/03/1997	2020202020	abc@gmail.com	Presc.

Big Show Nephrologist
Milkha Singh hematologist
narender yogi Dermatologists

About About About

profile_image profile_image profile_image

8.13 Dashboard–Doctor – Schedule

MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

Doctor's Dashboard

My Profile Patients Schedule Appointments Prescriptions

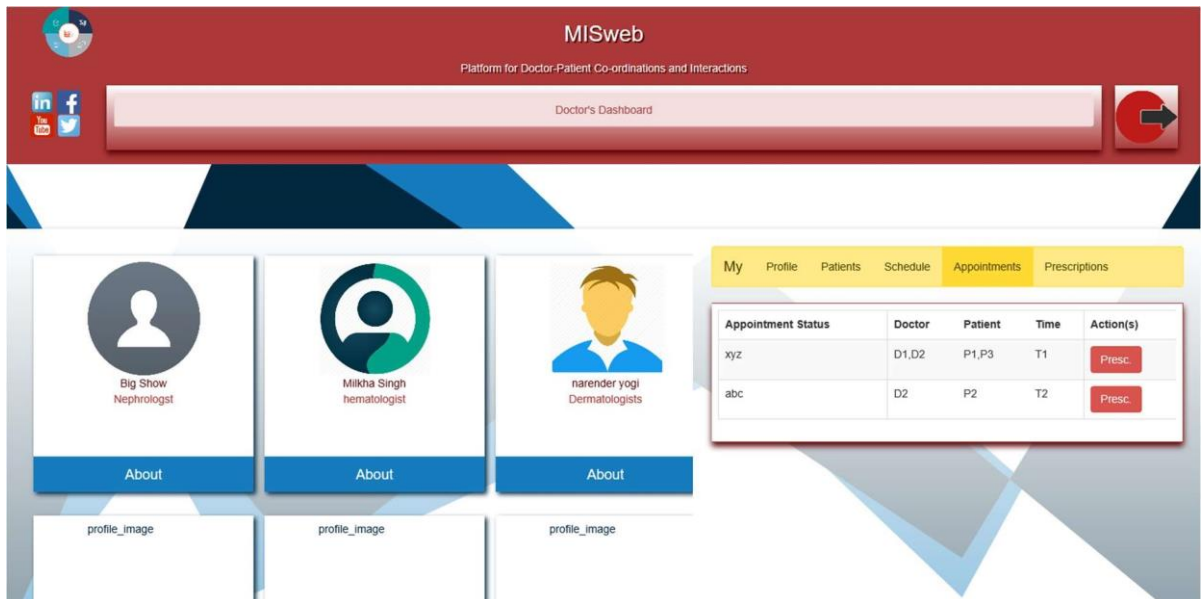
From Date	To Date	From Time	To Time	Next	Action(s)
12/02/2018	13/02/2018	12:50	2:00	2 months	Presc.
14/02/2013	15/02/2014	5:00	6:00	4 months	Presc.

Big Show Nephrologist
Milkha Singh hematologist
narender yogi Dermatologists

About About About

profile_image profile_image profile_image

8.14 Dashboard–Doctor – Appointments



MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

Doctor's Dashboard

My Profile Patients Schedule **Appointments** Prescriptions

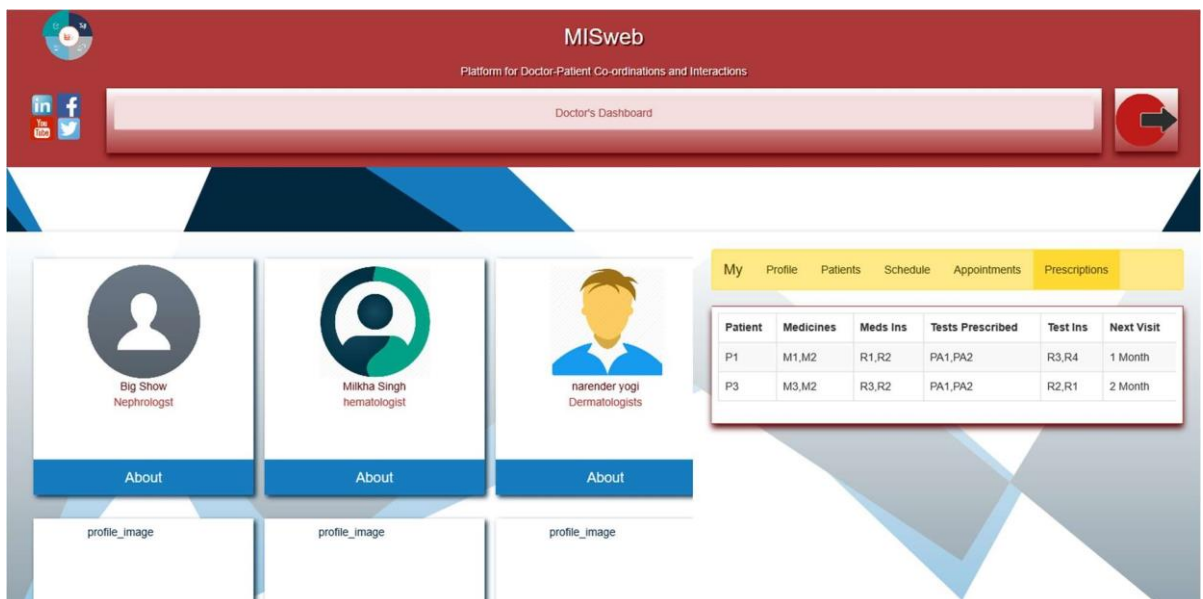
Appointment Status	Doctor	Patient	Time	Action(s)
xyz	D1,D2	P1,P3	T1	Presc.
abc	D2	P2	T2	Presc.

Big Show Nephrologist
Mikha Singh hematologist
narender yogi Dermatologists

About About About

profile_image profile_image profile_image

8.15 Dashboard–Doctor – Prescriptions



MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

Doctor's Dashboard

My Profile Patients Schedule Appointments **Prescriptions**

Patient	Medicines	Meds Ins	Tests Prescribed	Test Ins	Next Visit
P1	M1,M2	R1,R2	PA1,PA2	R3,R4	1 Month
P3	M3,M2	R3,R2	PA1,PA2	R2,R1	2 Month

Big Show Nephrologist
Mikha Singh hematologist
narender yogi Dermatologists

About About About

profile_image profile_image profile_image

8.16 Dashboard Expanded–Doctor – Profile

MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

My Dashboard - Medical Information System

Dashboard Profile Patients Schedule Appointments Prescriptions

Name Osama kashyap [Edit](#)

Qualification MBBS

Specialization cardiologists

Continue

8.17 Dashboard Expanded–Doctor – Patients

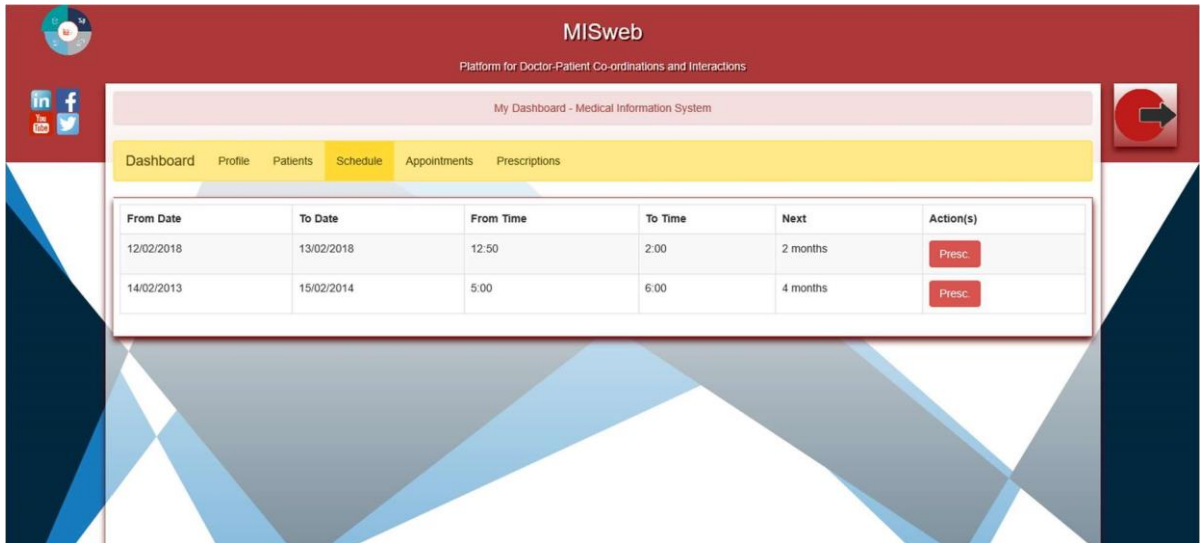
MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

My Dashboard - Medical Information System

Dashboard Profile Patients Schedule Appointments Prescriptions

Name	Vitals	Gender	DOB	Mobile	Email	Action(s)
Tiger Chaudary	xyz	M	13/12/2009	1010101010	xyz@gmail.com	Presc.
Bhalu Shukla	abc	M	15/03/1997	2020202020	abc@gmail.com	Presc.

8.18 Dashboard Expanded–Doctor – schedule



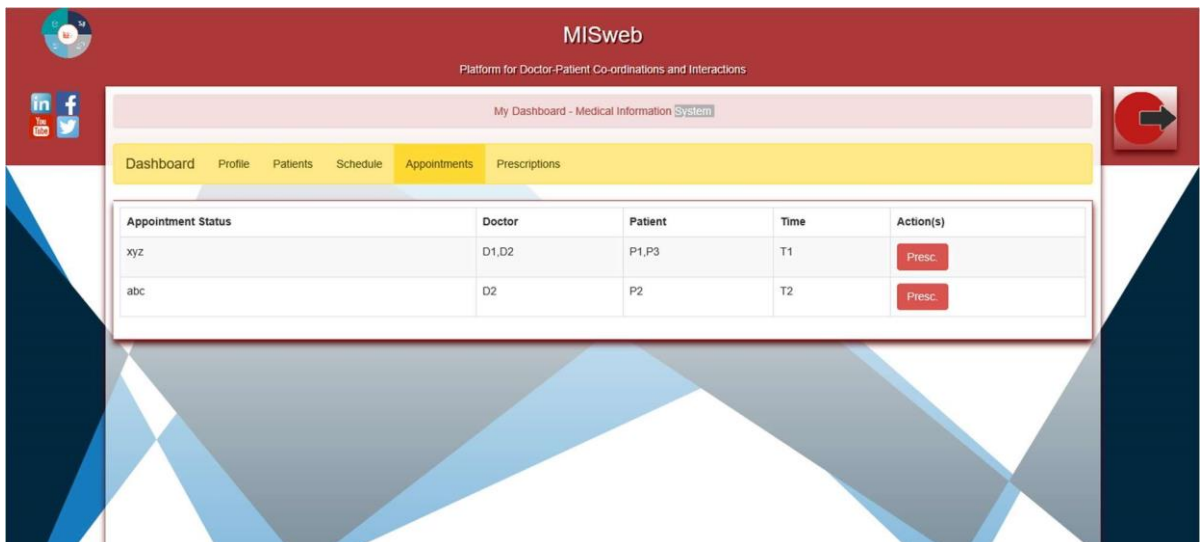
MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

My Dashboard - Medical Information System

Dashboard Profile Patients **Schedule** Appointments Prescriptions

From Date	To Date	From Time	To Time	Next	Action(s)
12/02/2018	13/02/2018	12:50	2:00	2 months	Presc.
14/02/2013	15/02/2014	5:00	6:00	4 months	Presc.

8.19 Dashboard Expanded–Doctor – Appointments



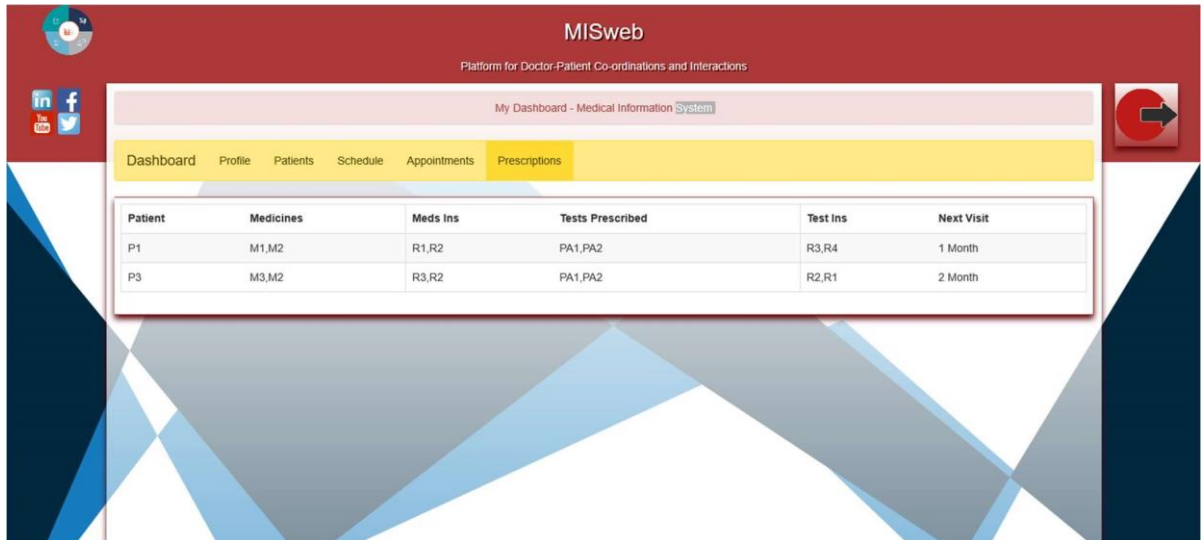
MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

My Dashboard - Medical Information System

Dashboard Profile Patients Schedule **Appointments** Prescriptions

Appointment Status	Doctor	Patient	Time	Action(s)
xyz	D1,D2	P1,P3	T1	Presc.
abc	D2	P2	T2	Presc.

8.20 Dashboard Expanded–Doctor – Prescriptions



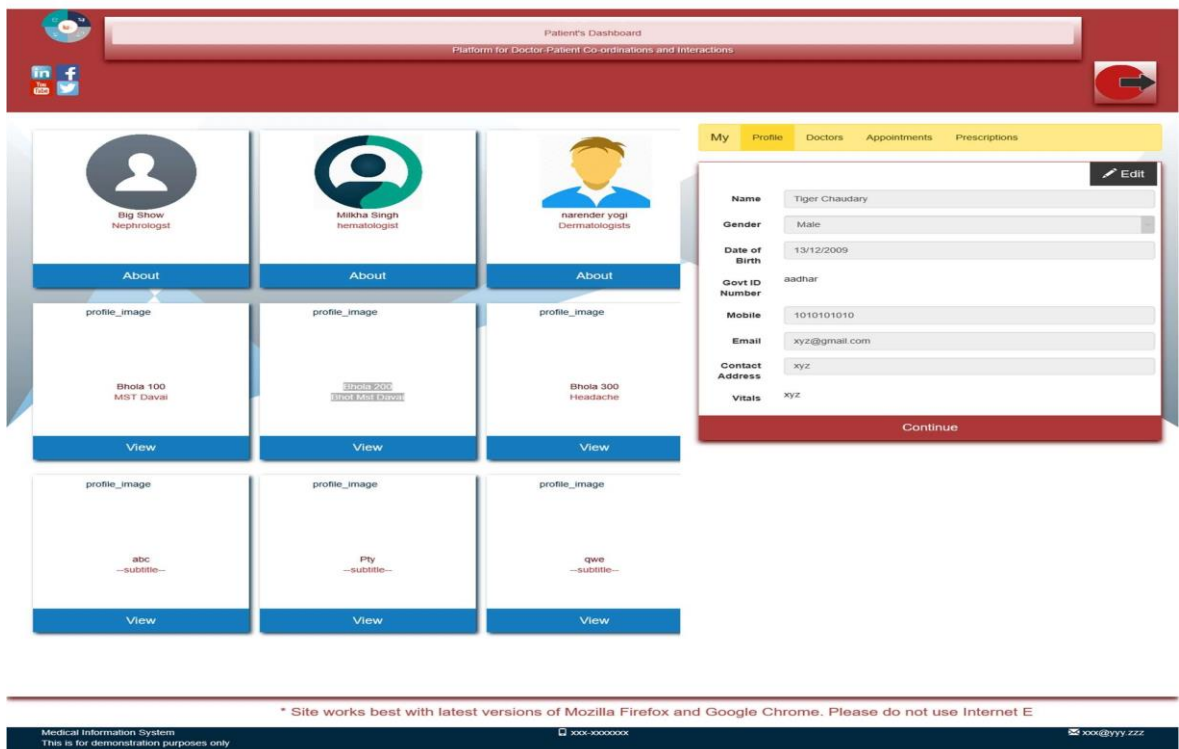
MISweb
Platform for Doctor-Patient Co-ordinations and Interactions

My Dashboard - Medical Information System

Dashboard Profile Patients Schedule Appointments Prescriptions

Patient	Medicines	Meds Ins	Tests Prescribed	Test Ins	Next Visit
P1	M1,M2	R1,R2	PA1,PA2	R3,R4	1 Month
P3	M3,M2	R3,R2	PA1,PA2	R2,R1	2 Month

8.21 Dashboard–Patient- Profile



Patient's Dashboard
Platform for Doctor-Patient Co-ordinations and Interactions

My Profile Doctors Appointments Prescriptions

Doctors:

- Big Show** Nephrologist
- Mikha Singh** hematologist
- narender yogi** Dermatologists

Profile Details:

Name: Tiger Chaudary

Gender: Male

Date of Birth: 13/12/2009

Govt ID Number: aadhar

Mobile: 1010101010

Email: xyz@gmail.com

Contact Address: xyz

Vitals: xyz

Continue

* Site works best with latest versions of Mozilla Firefox and Google Chrome. Please do not use Internet E

Medical Information System
This is for demonstration purposes only
Developed by g76

xxx-xxxxxxx

xxx@yyyy.zzz

8.22 Dashboard- Patient –Doctor

The screenshot shows the 'Patient's Dashboard' with the subtitle 'Platform for Doctor-Patient Co-ordinations and Interactions'. The dashboard features three doctor profiles on the left: Big Show (Nephrologist), Mikha Singh (hematologist), and narender yogi (Dermatologists). Each profile has an 'About' button and a 'profile_image' label. On the right, there is a navigation bar with tabs: My, Profile, Doctors, Appointments, and Prescriptions. The 'Doctors' tab is active, displaying a table of doctors.

Name	Qualification	Specialization	Action(s)
Osama kashyap	MBBS	cardiologists	Book Appointment
narender yogi	MBBS	Dermatologists	Book Appointment

8.23 Dashboard- Patient –Appointments

The screenshot shows the 'Patient's Dashboard' with the subtitle 'Platform for Doctor-Patient Co-ordinations and Interactions'. The dashboard features three doctor profiles on the left: Big Show (Nephrologist), Mikha Singh (hematologist), and narender yogi (Dermatologists). Each profile has an 'About' button and a 'profile_image' label. On the right, there is a navigation bar with tabs: My, Profile, Doctors, Appointments, and Prescriptions. The 'Appointments' tab is active, displaying a table of appointment status.

Appointment Status	Doctor	Patient	Time
xyz	D1,D2	P1,P3	T1

8.24 Dashboard- Patient – Prescriptions

The screenshot displays a 'Patient's Dashboard' with a red header bar. The header contains a circular logo on the left, a central title bar with the text 'Patient's Dashboard' and 'Platform for Doctor-Patient Co-ordinations and Interactions', and a red circular button with a white arrow on the right. Below the header, there are three doctor profile cards. Each card has a circular profile picture, the doctor's name and specialty, and an 'About' button. The first card is for 'Big Show Nephrologist', the second for 'Mikha Singh hematologist', and the third for 'narender yogi Dermatologists'. Below each card is a 'profile_image' label. To the right of the doctor cards is a yellow navigation bar with tabs: 'My', 'Profile', 'Doctors', 'Appointments', and 'Prescriptions'. The 'Prescriptions' tab is active. Below the navigation bar is a table with the following data:

Medicines	Meds Ins	Tests Prescribed	Test Ins	Next Visit
M3,M2	R3,R2	PA1,PA2	R2,R1	2 Month
M1,M2	R1,R2	PA1,PA2	R3,R4	1 Month

REFERENCES

- [1]. Ming-Hsuan Yang, David J. Kriegman, Narendra Ahuja, “Medical Information System: A Survey”, IEEE Transactions on Medical Systems, Volume. 24, No. 1, 2002.
- [2]. Paul Viola and Michael J. Jones, “Medical Information System:”, International Journal of Medical Systems: 57(2), p.p. 137–154, 2004.
- [3]. William Robson Schwartz, Huimin Guo, Jonghyun Choi, Larry S. Davis, “Medical Information System:”, IEEE Transactions On Clinic Management, Volume. 21, No. 4, 2012.
- [4]. Dayanand S. Shilwant, Dr. A.R. Karwankar, “Medical Information System:”, International Journal of Electronics, Communication & Soft Computing, Science and Engineering, ISSN 2277-9477, Volume 2, 2003.
- [5]. Matthew A. Turk and Alex P. Pentland, “Medical Information System:”, Medical Systems: 1991. Proceedings CVPR’91, IEEE Computer Society Conference, p.p. 586-591, 1991.