

# **MEDICAL INTERACTION CENTER**

*A project report Submitted to*

**MAHATMA GANDHI UNIVERSITY, KOTTAYAM**

*In partial fulfillment of the requirement for the award of the degree of*

**BACHELOR OF COMPUTER APPLICATIONS**

*Submitted by*

**RAKESH P S**  
**(Reg. No: 200021095935)**

*Under the supervision and guidance of*

**Ms. ASWATHY T C**

*(Assistant Professor)*



**DEPARTMENT OF  
COMPUTER APPLICATIONS**

**MARYGIRI COLLEGE OF ARTS AND SCIENCE**

**KOOTHATTUKULAM, ERNAKULAM**  
**(Affiliated To Mahatma Gandhi University, Kottayam)**

**(2020- 2023)**

**MARYGIRI COLLEGE OF ARTS AND SCIENCE**  
**KOOTHATTUKULAM, 686662**

**DEPARTMENT OF COMPUTER APPLICATIONS**



**CERTIFICATE**

This is to certify that the Project report titled “**MEDICAL INTERACTION CENTER**” submitted by **RAKESH P S** towards partial fulfillment of the requirements of the award of the degree of Bachelor of Computer Application is a record of bonafide work carried out by them during the academic year 2020-2023.

Signature of Guide

Signature of HOD

Dept. of Computer Application

Submitted for the viva-voice held on .....

**Internal Examiner**

**External Examiner**

## **DECLARATION**

I, **RAKESH P S** hereby declare that the project work entitled “**MEDICAL INTERACTION CENTER**” is a record of bona fide research carried out by us under the supervision and guidance of Assistant Professor Ms. ASWATHY T C Department of Computer Applications, Marygiri College of Arts and Science, Koothattukulam. I also declare that it has not been previously submitted for the award of any Degree, Diploma or similar titles by any University or similar other institutions.

Place: Koothattukulam

Date:

**RAKESH P S**

## **ACKNOWLEDGEMENT**

First and foremost, we thank Almighty God for His gracious guidance throughout the project. We acknowledge our deep sense of gratitude to **Dr. M V GEORGEKUTTY** the Principal for permitting us to do this project. We take the immense pleasure in expressing our thanks to Head of the Department **Ms. Aswathy T C** for her kind patronages in making this project a successful one.

I would like to extend my sincere thanks to Assistant Professor **Ms. Aswathy T C** for guidance and cooperation, without which this would not have been a success. This leaf of acknowledgement would not be complete without a special word of thanks to our beloved parents for their valuable support, encouragement and love, which enable us to successfully bring out this project.

**RAKESH P S**

## **ABSTRACT**

The establishment and improvement of doctor-patient interaction system is a very important requirement, especially when the communication technology is developing rapidly. The advantages of the web can be I useful to make up the time and distance between doctors and patients and to provide fast and adequate medical services. Through the connection between user terminals and specific service, both doctors and patients are able to obtain required data to achieve a better interaction.

The platform, Web services, and database technology are all gradually maturing so that we can develop a doctor- patient interaction system for Android to meet the needs of the patient and to provide Communication with patients by the doctor's more efficient and convenient means of communication with patients.

# CONTENTS

<b>1. INTRODUCTION</b>	<b>1</b>
1.1 BACKGROUND AND MOTIVATION	1
1.2 THE PROPOSED SYSTEM	1
1.3 PROJECT SCOPE	2
1.3.1 LIMITATIONS OF EXISTING SYSTEM	2
1.3.2 ADVANTAGES OF PROPOSED SYSTEM	3
1.3.3 DISADVANTAGES OF PROPOSED SYSTEM	3
<b>2. SYSTEM ANALYSIS</b>	<b>4</b>
2.1 INTRODUCTION	4
2.2 STAKE HOLDERS OF THIS PROJECT	5
2.2.1 ADMIN	5
2.2.2 DOCTOR	5
2.2.3 MEDICAL SHOP	5
2.2.4 DELIVERY BOY	6
2.2.5 CUSTOMER	6
2.3 SOFTWARE REQUIREMENT SPECIFICATIONS	6
2.3.1 ADMIN	6
2.3.2 DOCTOR	7
2.3.3 MEDICAL SHOP	7
2.3.4 DELIVERY BOY AND CUSTOMER	7
2.4 FEASIBILITY STUDY	8
2.5 HARDWARE AND SOFTWARE REQUIREMENTS	10
2.5.1 SOFTWARE SPECIFICATIONS	10
25.1.1 Python	10
25.1.2 MySQL	11
25.1.3 XAMPP SERVER	13
25.1.4 SUBLIME TEXT	14
25.1.5 ANDROID STUDIO	14
25.1.6 SQLyog	15
25.1.7 WINDOWS 11	15
25.1.8 MICROSOFT WORD	16
25.1.9 LUCIDCHART	17
2.5.2 HARDWARE REQUIREMENTS	17
<b>3. SYSTEM DESIGN</b>	<b>19</b>
3.1 SYSTEM ARCHITECTURE	19
3.2 MODULE DESIGN	20
3.3 DATABASE DESIGN	21
3.3.1 NORMALIZATION	22
3.3.2 TABLE STRUCTURE	22
3.3.3 DATAFLOW DIAGRAM	31
3.4 INTERFACE DESIGN	41

3.4.1	USER INTERFACE SCREEN DESIGN	41
3.4.2	OUTPUT DESIGN	42
<b>4.</b>	<b>IMPLEMENTATION</b>	<b>44</b>
4.1	CODING STANDARD	44
4.2	SAMPLE CODE	46
<b>5.</b>	<b>TESTING</b>	<b>51</b>
5.1.	TEST CASES	51
5.1.1.	UNIT TESTING	52
5.1.2.	INTEGRATION TESTING	54
5.1.3.	BLACK BOX TESTING	54
5.1.4.	WHITE BOX TESTING	55
5.1.5.	VALIDATION TESTING	56
5.1.6.	USER ACCEPTANCE TESTING	57
5.2.	TEST CASE DOCUMENTS	57
<b>6.</b>	<b>CONCLUSION</b>	<b>59</b>
6.1.	FUTURE ENHANCEMENTS	59
<b>7.</b>	<b>APPENDIX</b>	<b>60</b>
7.1.	REFERENCE	60
7.2.	SCREENSHOT	61

## **1. INTRODUCTION**

### **1.1 BACKGROUND AND MOTIVATION**

In today's world if someone wants to book a Doctor's Appointment, we need to call in a clinic or personally go to that place and book the appointment. This consumes the precious time of the patient. Also, if the doctor cancels his / her schedule, the patient does not come to know about it unless he/she goes to the clinic.

Big hospitals usually have huge number of staff and doctors. Managing the time, schedule and proper regularity of tasks becomes very difficult. In order to have efficient and effective system of doctor-patient interaction, a doctor patient interaction system is designed that only assists the doctors in their work but also helps patients in numerous ways like booking doctor appointments and assessing medical reports and progress of treatment. The portal permits the doctors to maintain their empty slots for booking by the patients using the same online system. Patients are requiring to look for the empty slots of doctors using the same online system and reserve the slots using the required information. The system maintains records of all the doctors and manages the appointment data of all the doctors for different times and dates. When a patient visits a doctor his/her date and time of visit is entered by the doctor in the database.

### **1.2 THE PROPOSED SYSTEM**

The proposed system consists of four panels: Doctor, Patient, Hospital or clinic, and Admin. The users will first have to download the application and install it on their mobile devices. Once installed, this application will remain into the device permanently until the user deletes it or uninstalls it. The patient will have to register into the application for the first time. On registering, the patient will receive a username and password. The patient can use this username and password for logging into the app each time he/she uses it. After logging in, the patient will have to select a filtration type.

The filtration is done on two bases: Gender wise and Specialty wise. After selecting the filtration type, the doctor's list will be displayed. The patient can select any particular doctor and view his profile. And patient gives reviews on doctor profile. Also, the patient can view the doctor's profile and look for an appointment. The patient will then send a request for an appointment. The doctor can either accept the appointment or reject it. The database will get updated accordingly and the patient will get a confirmation message. The add-on to this system



is that the patient will receive a notification 2 hours before the actual appointment. As well as, if a doctor cancels the appointment patient received a message for appointment cancellation.

This will be very useful in case the patient tends to forget the appointment. Also, the doctor can search patient history by using a unique ID.

## **1.3 PROJECT SCOPE**

### **1.3.1. Limitations of Existing System**

- If a doctor doesn't check the notification for confirmation which is booking by patient, then the main motto of this system will be failed.

- **Limited Patient Access**

Patients who lack access to technology or who are not comfortable with technology may not be able to benefit from telemedicine or other online consultation systems.

- **Limited Treatment Options**

While telemedicine can be useful for diagnosis and management of certain conditions, it may not be appropriate for all medical conditions or emergencies that require urgent attention.

- **Limited Physical Examination**

The lack of physical interaction between the doctor and patient limits the doctor's ability to perform a physical examination and obtain certain diagnostic information, such as blood pressure or reflexes.

### 132. Advantages of Proposed System

- Increased Access to Medical Care

Doctor-patient interaction systems can help patients in remote or underserved areas access medical care without the need for travel or time away from work

- Convenience

It can provide a convenient way for patients to access medical care, without the need for them to take time off work or travel to a medical facility.

- Improved Patient Satisfaction

The combination of doctor-patient interaction systems and home delivery of medicines can improve patient satisfaction by providing a convenient and accessible way for patients to access medical care and medications.

- Improved Medication Adherence expectations.

Home delivery of medicines can help improve medication adherence, as patients are more likely to take their medications as prescribed if they are delivered directly to their homes.

- Communication Facilities

It gives proper communication facilities to contact with both donor doctor and patient.

### 133. Disadvantages of Proposed system

- Limited Treatment Options

While telemedicine can be useful for diagnosis and management of certain conditions, it may not be appropriate for all medical conditions or emergencies that require urgent attention.

- Prescription Errors

The use of this system and home delivery of medicines can increase the risk of prescription errors, particularly if the patient's medical history or current medications are not accurately documented.

## 2. SYSTEM ANALYSIS

### 2.1 INTRODUCTION

Software Engineering is the analysis, design, construction, verification and management of technical or social entities. To engineer software accurately, a software engineering process must be defined. System analysis is a detailed study of the various operations performed by the system and their relationship within and module of the system. It is a structured method for solving the problems related to the development of a new system. The detailed investigation of the present system is the focal point of system analysis. This phase involves the study of parent system and identification of system objectives. Information has to be collected from all people who are affected by or who use the system. During analysis, data are collected on the variable files, decision point and transactions handled by the present system. The main aim of system is to provide the efficient and user-friendly automation. So, the system analysis process should be performed with extreme precision, so that an accurate picture of existing system, its disadvantages and the requirements of the new system can be obtained.

System analysis involves gathering the necessary information and using the structured tool for analysis. This includes the studying existing system and its drawback, designing a new system and conducting cost benefit analysis. System analysis is a problem-solving activity that requires intensive communication between the system users and system developers. The system is studied to the minute detail and analyzed. The system is viewed as a whole and the inputs to the system are identified. The outputs from the organization are traced through various phases of processing of inputs.

There are several different approaches to system analysis. When a computer-based information system is developed, systems analysis (according to the Waterfall model) would constitute the following steps:

- The development of a feasibility study, involving determining whether a project is economically, technologically, and operationally feasible.
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
- Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for and so on.

Techniques such as interviews, questionnaires etc. can be used for the detailed study of these processes. The data collected by these sources must be scrutinized to arrive at a conclusion. The conclusion is an understanding of how the system functions. This system is called the Existing System. The Existing system is then subjected to close observation and the problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as a proposal which is the Proposed System. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is then presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is a loop that ends as soon as the user is satisfied with the proposal.

## **2.2 STAKE HOLDERS OF THIS PROJECT**

### **2.2.1. Admin**

Admin plays the major role in the system. Admin manages all the jobs performed and has the provision to view and access all details. It also has the provision to control the Doctor, Patient, Medical shops, Delivery boy. Admin can enquire about any of the issues. Admin can check all the functioning of system. Admin can approve the Doctor and approved Doctor can only login to the system. Admin can enquire about the Patient account if he/she fails to login to this system. Admin can add or remove another admin into it or from the system. Admin is possible in to update username and password.

### **2.2.2. Doctor**

The Doctors are the most important stakeholder in this project. The Doctor also has to register with their basic details and the admin will approve if he/she is genuine else he/she will be rejected. The role of the doctor in a doctor-patient interaction system is to diagnose and treat the patient's condition, communicate effectively with the patient, educate them on their condition and treatment options, monitor their progress, and act as an advocate for their health and well-being. The doctor's primary responsibility is to provide high-quality medical care and support the patient's overall health.

### **2.2.3. Medical Shop**

The role of a medical shop is to provide the patient with access to the medication and medical supplies that the doctor has prescribed. The medical shop may also provide additional health-related products, such as over-the-counter medications, first aid supplies, and health

supplements. The medical shop plays an important role in ensuring that the patient receives the medication and supplies needed to manage their condition effectively. Overall, the role of the medical shop in the doctor-patient interaction system is to provide convenient and timely access to the medication and supplies needed to support the patient's health and well-being.

#### **2.2.4. Delivery Boy**

Delivery boy is to transport medication and medical supplies from the medical shop to the patient's location. The delivery boy plays a crucial role in ensuring that the patient receives their medication and supplies in a timely and efficient manner. The delivery boy may also provide additional support, such as helping the patient to understand how to use a particular medication or medical supply. Overall, the role of the delivery boy in the doctor-patient interaction system is to ensure that the patient receives the care and support they need to manage their health effectively.

#### **2.2.5. Customer**

The customer, or patient, is to actively participate in their own healthcare by communicating their symptoms, concerns, and expectations to the doctor. The patient plays an important role in the healthcare process by providing the doctor with accurate and detailed information about their health, following the treatment plan prescribed by the doctor, and providing feedback on the effectiveness of the treatment. Additionally, the patient may be responsible for obtaining and taking their medication as prescribed, keeping track of their symptoms and side effects, and attending follow-up appointments.

### **2.3 SOFTWARE REQUIREMENT SPECIFICATION**

#### **2.3.1. Admin**

- The system shall allow a provision to login the administrator by entering username and password.
- The system shall provide the provision to verify the user accounts and provide support for the system.
- The system should provide the provision for admin to verify the doctor, medical shop, delivery boy, customer or patient's account.
- Admin can provide the provision to add new medical shops.
- The system should provide the provision to display the count of doctors and medical shops available.
- The system should provide the provision to display the count of doctors on the basis of

- The system should provide the provision for admin to either accept or reject the newly joined Doctors.

### **2.3.2. Doctor**

- The system shall allow the provision for doctors to login by entering username and password.
- Doctors will have a verification under admin. Newly joined doctor which are approved by admin can only login to the system using username and password.
- Doctors Can view scheduled appointments.
- Doctors can upload prescription in this system to get medicines to users.
- The request of doctors received by the admin can either accept or reject the request.

### **2.3.3. Medical Shops**

- The medical shop shall be able to login with his/her username and password, also he/she shall be able to logout.
- Can view the patient's doctor prescriptions.
- Can view and update the information of the patient's payment status.
- Should allow the medical shop to communicate with the delivery boy to coordinate deliveries and ensure timely delivery of medication and supplies.

### **2.3.4. Delivery Boy and Customer**

- The patients or customers shall be able to login with his/her username and password, also he/she shall be able to logout.
- The patient shall be able to schedule appointment.
- The Delivery boys shall be able to login with his/her username and password, also he/she shall be able to logout.
- The patient shall be able to view his/her doctor's prescription.
- Should allow the delivery boy to communicate with the medical shop or pharmacy to coordinate deliveries and ensure timely delivery of medication and supplies.

## 2.4 FEASIBILITY STUDY

Feasibility is defined as the practical extent to which a project can be performed successfully. To evaluate feasibility, a feasibility study is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study. The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards. Various other objectives of feasibility study are listed below.

- To analyze whether the software will meet organizational requirements.
- To determine whether the software can be implemented using the current technology and within the specified budget and schedule.
- To determine whether the software can be integrated with other existing software.

### Technical Feasibility

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

- Analyses the technical skills and capabilities of the software development team members.
- Determines whether the relevant technology is stable and established.
- Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

From our perspective there are two languages PHP, HTML and database MySQL which are used to develop this web-based applications. PHP is used in the front end and MySQL is used in the back end. The Word to the Wise is web based and thus can be accessed through any browsers. As we are using these latest technologies which are currently trending and used by a number of developers across the globe, we can say that our project is technically feasible.

### **Operational Feasibility**

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority.
- Determines whether the solution suggested by the software development team is acceptable.
- Analyses whether users will adapt to a new software.
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

We found that our project will be satisfied for the client since we were discussing every detail about the software with the client at every step. The most important part of operational feasibility study is the input from client. So, the software is built completely according to the requirements of the client. We have used the current industry standards for the software. Hence, we can say that this software is operationally feasible.

### **Economic Feasibility**

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization.
- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis).
- Cost of hardware, software, development team, and training.

It is estimated that our project is economically feasible as development cost is very minimal since the tools and technologies used are available online. It's a group student project



so there are no personnel costs. Development time is well planned and will not affect other operations and activities of the individuals. Once the system has been developed, the companies purchasing the system will be providing with a manual for training purposes. There is no need to purchase new hardware since the existing computers can still be used to implement the new system.

## **2.5. HARDWARE AND SOFTWARE REQUIREMENTS.**

### **2.5.1. Software Specification**

This project is built upon the latest technology

- **Software Frontend:** HTML, CSS, JS, Python (For web application)
- **Software Backend:** Angular JS, PHP, MYSQL
- **Database:** MySQL
- **Web server:** XAMPP server
- **Web browser:** Internet Explorer/Google Chrome/firefox(for web application)
- **Operating System:** Windows 11

#### **2.5.1.1. Python**

Python is a high-level, interpreted programming language that was first released in 1991. It was created by Guido van Rossum, and has since become one of the most popular programming languages in use today. Python is known for its simplicity, readability, and ease of use, making it a great choice for beginners and experts alike.

Python is an object-oriented language that emphasizes code readability and clarity, and its syntax is designed to be easy to read and write. It has a large and active community of developers who have contributed to its growth and development over the years, and it is supported on a wide range of platforms, including Windows, macOS, Linux, and many others.

One of the major advantages of Python is its versatility. It can be used for a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, machine learning, game development, and more. Its ease of use and wide range of libraries and frameworks make it a popular choice for these applications.

Another advantage of Python is its ease of learning. Its syntax is designed to be easy to read and understand, making it a great choice for beginners who are just starting to learn programming. Additionally, there are many online resources, tutorials, and communities available to help new users get started with Python.

Overall, Python is a powerful and flexible programming language that is well-suited for a wide range of applications, and its popularity and active community make it a great choice for both beginners and experienced developers alike.

### **2.6.1.2. MySQL**

MySQL is the world's most popular opensource database software, with over 100 million copies of its software downloaded or distributed throughout its history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume Web sites, critical business systems, and packaged software — including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com.

The flagship MySQL offering is MySQL Enterprise, a comprehensive set of production-tested software, proactive monitoring tools, and premium support services available in an affordable annual subscription.

MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing opensource enterprise software stack. More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform locking.

MySQL was originally founded and developed in Sweden by two Swedes and a Finn: David Ax mark, Allan Larsson and Michael "Monty" Wideners, who had worked together since the 1980's. MySQL, the most popular open-source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

MySQL is a database management system. A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications. MySQL databases are relational. A relational database stores data in separate tables rather than putting all the data in

one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and pointers between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of —MySQL stands for —Structured Query Language. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, —SQL-92 refers to the standard released in 1992, —SQL:1999 refers to the standard released in 1999, and —SQL:2003 refers to the current version of the standard.

We use the phrase —the SQL standard to mean the current version of the SQL Standard at any time. MySQL software is Open Source. Open-Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs.

The MySQL software uses the GPL (GNU General Public License), <http://www.fsf.org/licenses/>, to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL need to embed MySQL code into a commercial application, you can buy a commercially licensed version.

The MySQL Database Server is amazingly fast, reliable, scalable, and easy to use. If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments

for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet. MySQL Server works in client/server or embedded systems.

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

A large amount of contributed MySQL software is available. MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the MySQL Database Server.

#### **2.6.1.3. XAMPP Server**

XAMPP is a popular software package that allows users to create a local web server environment on their computer. It is an open-source software package that is available for Windows, Mac, and Linux operating systems. XAMPP stands for Cross-Platform, Apache, MySQL, PHP, and Perl. These are the components that make up the software package.

XAMPP includes Apache web server, MySQL database, PHP, and Perl programming languages. It also includes additional tools and modules such as phpMyAdmin, OpenSSL, FileZilla FTP server, and Mercury Mail server. These tools are essential for web development, testing, and deployment.

One of the major advantages of XAMPP is that it allows developers to create a local web server environment on their own computer, which can be used to test and develop websites and web applications before they are deployed to a live server. This can save a lot of time and effort compared to developing and testing on a remote server.

XAMPP is also easy to install and configure, making it a great choice for beginners who are just starting to learn web development. It includes a simple graphical user interface that allows users to start and stop the server, manage databases, and configure settings easily. Overall, XAMPP is a powerful and popular software package that is essential for web development and testing. It provides a complete and easy-to-use environment for creating and testing web applications on a local server, and it is a great choice for beginners and experienced developers alike.

#### **2.6.1.4. Sublime Text**

Sublime Text is a popular code editor used by developers for writing code, markup, and prose. It is a lightweight and customizable text editor with a sleek user interface and powerful features that make it a popular choice for web developers.

One of the key features of Sublime Text is its ability to handle multiple files and projects at once, making it easy to switch between files and manage multiple projects simultaneously. It also has a powerful search and replace function that allows users to quickly find and replace text in their code.

Sublime Text supports many programming languages and markup formats, including HTML, CSS, JavaScript, Python, and more. It has a built-in package manager that allows users to easily install and manage extensions and plugins to add functionality to the editor.

Another popular feature of Sublime Text is its "Go to Anything" command, which allows users to quickly jump to a specific line or file in their code using a keyboard shortcut. It also has a powerful autocomplete function that suggests code snippets and syntax as users' type.

Sublime Text is available for Windows, Mac, and Linux operating systems and has a free evaluation version with no time limit. Users can purchase a license for continued use and support. Overall, Sublime Text is a powerful and customizable code editor with many features that make it a popular choice for web developers. Its lightweight design, powerful search and replace function, support for multiple languages, and customizable interface make it a top choice for developers looking for a powerful text editor.

#### **2.6.1.5. Android Studio**

Android Studio is an integrated development environment (IDE) created by Google for developing Android applications. It is built on top of the popular JetBrains IntelliJ IDEA software, and provides developers with a powerful set of tools to develop Android apps.

One of the key features of Android Studio is its layout editor, which allows developers to create user interfaces using a drag-and-drop interface. The layout editor supports multiple screen sizes and orientations, making it easy to create responsive UIs.

Android Studio also comes with a powerful code editor, which includes features such as syntax highlighting, code completion, and refactoring tools. It supports multiple programming languages, including Java, Kotlin, and C++, and provides developers with a powerful set of debugging tools to identify and fix bugs.

In addition to its code editor and layout editor, Android Studio comes with a built-in emulator, which allows developers to test their apps on a virtual Android device. This makes it easy to test apps on different screen sizes and versions of Android.

Android Studio also provides developers with tools to manage their app's dependencies and to package and deploy their apps to the Google Play Store. It supports integration with popular version control systems, such as Git and GitHub, and provides developers with tools to manage their app's releases and versioning.

Overall, Android Studio is a powerful and comprehensive development environment for building Android apps. Its layout editor, code editor, debugging tools, emulator, and deployment tools make it a top choice for Android app developers.

#### **2.6.1.6. SQLyog**

SQLyog is a popular MySQL database management tool that provides a graphical user interface (GUI) for managing MySQL databases. It is available for Windows and Linux operating systems, and provides a range of features that simplify database administration and development.

SQLyog allows users to connect to and manage multiple MySQL servers from a single interface, making it easy to work with multiple databases and servers. It provides a range of features for database administration, including creating and managing databases, tables, indexes, and users, and performing backup and restore operations.

One of the key features of SQLyog is its visual query builder, which allows users to build complex SQL queries using a drag-and-drop interface. It also provides a code editor with syntax highlighting, code completion, and debugging tools, making it easy to write and debug SQL queries.

SQLyog also includes tools for importing and exporting data between databases, generating schema and data comparison reports, and scheduling database backup and optimization tasks.

Overall, SQLyog is a powerful MySQL database management tool that provides a range of features for database administration and development. Its intuitive user interface and visual query builder make it easy to work with MySQL databases, and its advanced features and automation tools make it a top choice for database administrators and developers.

#### **2.6.1.7. Windows 11**

Operating System is defined as a program that manages the computer hardware. An operating system can be viewed as a scheduler, where it has resources for which it has charge. Resources include CPU, memory, I/O device, and disk space. In another view, the operating system is a new machine. The third view is that operating system is a multiplexer which allows sharing of resources provides protection from interference and provides a level of cooperation between users. This project is developed using Windows 11 as the operating system and supports its latest versions. Windows 11, the first major Windows release since 2015, builds upon its predecessor by revamping the user interface to follow Microsoft's new fluent design

guidelines. The redesign, which focuses on ease of use and flexibility, comes along side new productivity and social features and updates to security and accessibility, addressing some of the deficiencies of windows 10.

A redesigned user interface is present frequently throughout the operating system, building upon fluent design system; translucency, shadows, a new color palette, and rounded geometry are prevalent throughout the UI. A prevalent aspect of the design is an appearance known as "Mica", described as an "opaque, dynamic material that incorporates theme and desktop wallpaper to paint the background of long-lived windows such as apps and settings". Much of the interface and start menu takes heavy inspiration from the now canceled Windows 10X.<sup>1</sup> The Segoe-UI font used since window vista has been updated to a variable version, improving its ability to scale between different display resolutions.

#### **2.6.1.8. Microsoft Word**

Microsoft Word (or simply Word) is a word processor developed by Microsoft. It was first released on October 25, 1983 under the name Multi-Tool Word for Xenix systems. Subsequent versions were later written for several other platforms including IBM PCs running DOS (1983), Apple Macintosh running the Classic Mac OS (1985), AT&T Unix PC (1985), Atari ST (1988), OS/2 (1989), Microsoft Windows (1989), SCO Unix (1994), and macOS (formerly OS X; 2001).

Commercial versions of Word are licensed as a standalone product or as a component of Microsoft Office, Windows RT or the discontinued Microsoft Works suite. Unlike most MSDOS programs at the time, Microsoft Word was designed to be used with a mouse. Advertisements depicted the Microsoft Mouse, and described Word as a WYSIWYG, windowed word processor with the ability to undo and display bold, italic, and underlined text, although it could not render fonts. It was not initially popular, since its user interface was different from the leading word processor at the time, WordStar. However, Microsoft steadily improved the product, releasing versions 2.0 through 5.0 over the next six years. In 1985, Microsoft ported Word to the classic Mac OS (known as Macintosh System Software at the time). This was made easier by Word for DOS having been designed for use with high-resolution displays and laser printers, even though none were yet available to the general public. Following the precedents of Lisa Write and MacWrite, Word for Mac OS added true WYSIWYG features. It fulfilled a need for a word processor that was more capable than MacWrite. After its release, Word for Mac OS's sales were higher than its MS-DOS counterpart for at least four years.

### **2.6.1.9. LucidChart**

LucidChart is a diagram tool used to make flowcharts, organization charts, mind maps, project charts, and other business visuals. LucidChart has two versions: an online edition and a downloadable edition for Windows desktop.

LucidChart integrates with Microsoft Office products including Word, PowerPoint, and Excel and G Suite applications like Google Docs and Google Sheets. LucidChart has apps for Atlassian's Confluence, Jira, and Trello. Smart Draw is compatible with Google Drive, Dropbox, Box, and OneDrive.

Since 1994, the mission of LucidChart Software has been to expand the ways in which people communicate so that we can clearly understand each other, make informed decisions, and work together to improve our businesses and the world. We accomplish this by creating software and services that make it possible for people to capture and present information as visuals, while being a pleasure to use in 2019, we took this to the next level by launching Visual Script, which makes it easy to visualize data in relational formats like trees, flows, and timelines, automatically, without any human input. Visual Script is a relationship visualization platform that empowers organizations to visualize data across siloed ecosystems and gain critical insights in real-time. Today, LucidChart Software is one of the most sophisticated digital marketing organizations in the world with over 90,000 unique visitors to our website each business day and in excess of 3,000,000 installations of our apps each year. Smart Draw is used by more than half of the Fortune 500 and by over 250,000 public and private enterprises of all sizes around the world. Privately held, LucidChart Software is headquartered in San Diego, California.

### **2.6.2 Hardware requirements**

The selection of hardware configuring is a very task related to the software development, particularly inefficient RAM may affect adversely on the speed and corresponding on the efficiency of the entire system. The processor should be powerful to handle all the operations. The hard disk should have the sufficient to solve the database and the application.



**Hardware used for development**

CPU: Intel i5 Processor

Memory: 8 GB

Cache: 6 MB

Monitor: 15.6" Monitor

Keyboard: Standard 108 keys Enhanced Keyboard

Mouse: Optical Mouse

**Minimum Hardware Required for Implementation**

CPU: IV Processor

Memory: 256 MBA above

Cache: 512 Above

Hard Disk: 20 GB Above

Keyboard: Any

Mouse: Any

### **3. SYSTEM DESIGN**

#### **3.1 SYSTEM ARCHITECTURE**

A system architecture or system's architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures of the system.

System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g., the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs).

The system architecture can best be thought of as a set of representations of an existing (or to be created) system. It is used to convey the informational content of the elements comprising a system, the relationships among those elements, and the rules governing those relationships. The architectural components and set of relationships between these components that architecture describes may consist of hardware, software, documentation, facilities, manual procedures, or roles played by organizations or people. System architecture is primarily concerned with the internal interfaces among the system's components or subsystems, and the interface between the system and its external environment, especially the user.

The structural design reduces complexity, facilitates change and result in easier implementation by encouraging parallel development of different parts of the system. The procedural design transforms structural elements of program architecture into a procedural description of software components. The architectural design considers architecture as the most important functional requirement. The system is based on the three-tier architecture.

The first level is the user interface (presentation logic), which displays controls, receives and validates user input. The second level is the business layer (business logic) where the application specific logic takes place. The third level is the data layer where the application information is stored in files or database. It contains logic about to retrieve and update data. The important feature about the three-tier design is that information only travels from one level to an adjacent level.

### 3.2. MODULE DESIGN

Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality. Conceptually, modules represent a separation of concerns, and improve maintainability by enforcing logical boundaries between components. Different modules of this project include

#### 1. Admin

- ❖ Login
- ❖ Manage Type
- ❖ Manage Shops
- ❖ View Medicines
- ❖ View Doctors
  - Accept/Reject
- ❖ View Feedback

#### 2. Doctor

- ❖ Register
- ❖ Login
- ❖ View Appointment
  - View Customer Details
  - Make a Call
  - Upload Prescription

#### 3. Medical Shop

- ❖ Login
- ❖ Manage Medicine
- ❖ Update Stock
- ❖ View doctor's Prescription
  - Update medicine with rate
  - View Payed
- ❖ Update status to ready to dispatched

#### 4. Delivery Boys

- ❖ Register
- ❖ Login
- ❖ View any delivery's
  - View Place and details
  - Update status to delivered

#### 5. Customers

- ❖ Registration

- ❖ Login
- ❖ View Doctors
  - Make Appointment
- ❖ View Appointment
  - Make a Call
- ❖ View Medical Shop
  - Upload Doctor Prescription details
- ❖ View uploaded medicine details
  - Accept or Reject
  - Payment
  - View Status
- ❖ Add Feedback

### 3.3. DATABASE DESIGN

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy, quick, inexpensive and flexible for the users. The general theme behind a database is to integrate all information. Database design is recognized as a standard of management information system and is available virtually for every computer system. In database design several specific objectives are considered:

- Ease of learning and use
- Controlled redundancy
- Data independence
- More information at low cost
- Accuracy and integrity
- Recovery from failure
- Privacy and security
- Performance

A database is an integrated collection of data and provides centralized access to the data. Usually, the centralized data managing the software is called RDBMS. The main difference between RDBMS and other DBMS is the separation of data as seen by the program and data has in direct access to store device. This is the difference between logical and physical data.

### 3.3.1 Normalization

The term normalization refers to the way data items are grouped together into record structures. In other words, normalization is a technique of separating redundant fields and breaking up large tables into smaller ones. After the conceptual level, the next level of process of database design to organize the database structure into a good shape called Normalization. Normalization is adopted to overcome drawbacks like repetition of data, loss of information, Inconsistence. In our design, all tables have been normalized up to the third normal form. The different normal forms applied during the database design are given below.

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

#### FIRST NORMAL FORM

A relation is said to be in 1NF if it satisfies the constraints that it contains primary key.

#### SECOND NORMAL FORM

A relation is said to be in 2NF if and only if it satisfies all 1NF conditions for the primary key and every non-primary key attribute of the relation is fully dependent on its primary key alone. If a non-key attribute is not dependent on the key, it should be removed from the relation and placed as a separate relation i.e. the fields of the table in 2NF are all related to primary key.

#### THIRD NORMAL FORM

A relation is said to be in 3NF if only if only it is in 2NF and more over non-key attributes of the relation should not depend on other non-key attributes. The data in the system has to be stored and retrieved from database.

### 3.3.2 Table Structure

Table is a collection of complete details about a particular subject. These data are saved in rows and Columns. The data of each Row are different units. Hence, rows are called RECORDS and Columns of each row are called FIELDS.

Data is stored in tables, which is available in the backend the items and data, which are entered in the input, form id directly stored in this table using linking of database. We can link more than one table to input forms. We can collect the details from the different tables to display on the output.

There are mainly 15 tables in the project. They are,

1. users
2. login
3. medicalshop
4. deliveryboy
5. type
6. medicines
7. uploadprescription
8. medicinedetails
9. payment
10. feedback
11. delivery
12. doctors
13. appointments
14. prescription
15. rating

Table: **users**

Description: This table is used to store the details of customers or patients.

Field name	Datatype	Constraints	Description
User_id	int(11)	Not Null Primary key Auto increment	Unique id for user
Login_id	int(11)	Not Null	Unique Id when login
firstname	varchar(100)	Not Null	First Name
lastname	varchar(100)	Not Null	Last Name
dob	Varchar(50)	Not Null	Date Of Birth
phone	varchar(50)	Not Null	Phone number
email	varchar(30)	Not Null	Email of user
place	varchar(30)	Not Null	Name of place
district	varchar(30)	Not Null	Name of district

Table: **login**

Description: This table is used to store details of when anyone login.

Field name	Data type	Constraints	Description
Login_id	Int(11)	Not Null Primary key Auto increment	Unique id when login
username	varchar(100)	Not Null	Username of user
password	varchar(50)	Not Null	Password of user
usertype	Varchar(50)	Not Null	Identify which type of user login

Table: **medicalshop**

Description: This table is used to store the details of medical shop.

Field name	Data type	Constraints	Description
Medicalshop_id	Int(11)	Not Null Primary key Auto increment	Unique id of medical shop
Login_id	varchar(30)	Not Null	Login id of user
shopname	varchar(50)	Not Null	Name of shop
place	Varchar(50)	Not Null	Place of medical shop
city	Varchar(50)	Not Null	City of medical shop
district	Varchar(50)	Not Null	District of medical shop
email	Varchar(50)	Not Null	Email of medical shop
phone	Varchar(50)	Not Null	Phone number of medical shop



Table: **deliveryboy**

Description: Stores the details of delivery boy

Field name	Datatype	Constraints	Description
boy_id	int(11)	Not Null Primary key Auto increment	Unique id for delivery boy
Login_id	int(11)	Not Null	Unique Id when login
firstname	varchar(100)	Not Null	First Name
lastname	varchar(100)	Not Null	Last Name
phone	varchar(50)	Not Null	Phone number of delivery boy
email	varchar(30)	Not Null	Email of user

Table: **type**

Description: This table is used to store the details of which type of user is this .

Field name	Data type	Constraints	Description
type_id	Int(11)	Not Null Primary key Auto increment	Unique id user
type	varchar (20)	Not Null	Identify which type of user login

Table: **medicines**

Description: This table is used to store details of medicines

Field name	Data type	Constraints	Description
medicine_id	int (11)	Not Null Primary key Auto increment	Unique ID of medicine
Medicalshop_id	int (11)	Not Null	Unique id of medical shop
type_id	Int (11)	Not Null	Type id of user
medicine	varchar (50)	Not Null	Name of medicine
details	varchar (50)	Not Null	Details of medicine
rate	varchar (50)	Not Null	Rate of medicine
quality	varchar (50)	Not Null	quality
mdate	varchar (50)	Not Null	
edate	varchar (50)	Not Null	Expiry date of medicine

Table: **upload prescription**

Description: This table is used to store the details of prescriptions

Field name	Data type	Constraints	Description
Prescription_id	int (11)	Not Null Primary key Auto increment	Unique id of prescription
medicalshop_id	int (11)	Not Null	Unique id of medical shop
User_id	Int(11)	Not Null	Unique id user
uploadfile	Varchar(50)	Not Null	File upload
totalamount	Varchar(50)	Not Null	Total amount
date	Varchar(50)	Not Null	Date of prescription
status	Varchar(50)	Not Null	Status of prescription

Table: **medicinedetails**

Description: This table is used to store details of medicine

Field name	Data type	Constraints	Description
detail_id	int(11)	Not Null Primary key Auto increment	Unique id of medicine details
prescription_id	int(11)	Not Null	Unique id of prescription
Medicine_id	Int(11)	Not Null	Unique id of medicine
quantity	Varchar(50)	Not Null	Quantity of medicines
rate	Varchar(50)	Not Null	Rate of medicine
total	Varchar(50)	Not Null	Total of medicines

Table: **payment**

Description: This table is used to store details of payment

Field name	Data type	Constraints	Description
payment_id	int(11)	Not Null Primary key Auto increment	Unique id of payments
prescription_id	int(11)	Not Null	Unique id of prescription
amount	Varchar(50)	Not Null	Amount of medicines
rate	Varchar(50)	Not Null	Rate of medicine

Table: **feedback**

Description: This table is used to store feedback

Field name	Data type	Constraints	Description
feedback_id	int(11)	Not Null Primary key Auto increment	Unique id of feedback
User_id	Int(11)	Not Null	Unique id of user
feedback	Varchar(50)	Not Null	feedback
date	varchar(20)	Not Null	Date of feedback

Table: **delivery**

Description: This table is used to store delivery

Field name	Data type	Constraints	Description
Delivery_id	int(11)	Not Null Primary key Auto increment	Unique id of delivery
Prescription_id	int(11)	Not Null	Unique id of prescription
Boy_id	int(11)	Not Null	Unique id of delivery boy
date	Varchar(50)	Not Null	Date of delivery

Table: **doctors**

Description: This table is used to store details of doctors

Field name	Datatype	Constraints	Description
doctor_id	int(11)	Not Null Primary key Auto increment	Unique id of doctor
Login_id	int(11)	Not Null	Unique Id when login
fname	varchar(100)	Not Null	First Name
lname	varchar(100)	Not Null	Last Name
place	Varchar(100)	Not Null	Place of doctor
phone	varchar(100)	Not Null	Phone number of doctor
email	Varchar(100)	Not Null	Email of doctor

**Table: appointments**

Description: This table is used to store the appointment details

Field name	Data type	Constraints	Description
appointment_id	int(11)	Not Null Primary key Auto increment	Unique id of the appointments
Doctor_id	Int(11)	Not Null	Unique id of the doctors
user_id	date	Not Null	Id of the user
time	varchar(100)	Not Null	Time of appointment
date	varchar(100)	Not Null	Date of appoinmtent
status	varchar(100)	Not Null	status

**Table: prescription**

Description: This table is used to store prescription details

Field name	Datatype	Constraints	Description
pres_id	int(11)	Not Null Primary key Auto increment	Unique id of prescription
appoinmentment_id	int(11)	Not Null	Unique Id of appointment
file	varchar(100)	Not Null	files
date	varchar(100)	Not Null	Date of prescription
status	Varchar(100)	Not Null	status

**Table: rating**

Description: This table is used to store rating

Field name	Data type	Constraints	Description
rating_id	int(11)	Not Null Primary key Auto increment	Unique id of rating
boy_id	int(11)	Not Null	Unique id of delivery boy
user_id	Int(11)	Not Null	Unique id of user
rated	Varchar(50)	Not Null	rating
date	Varchar(50)	Not Null	Date of rating

### 3.3.3. Data Flow Diagram

#### 3.3.3.1 Introduction to Data Flow Diagrams

Data Flow Diagram is a network that describes the flow of data and processes that change, or transform, data throughout the system. This network is constructed by use a set of symbols that do not imply a physical implementation. It is a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. Data in files may also flow to processes as inputs. There are various symbols used in a DFD. Bubbles represent the processes.

Named arrows indicate the data flow. External entities are represented by rectangles. Entities supplying data are known as sources and those that consume data are called This network is constructed by use a set of symbols that do not imply a physical implementation. It a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. External that entities are represented by rectangles. Entities supplying data are known as sources and those consume data are called sinks. Data are stored in a data store by a process in the system. It is a graphical tool for structured analysis of the system requirements. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates, output-data-flows which go to other processes or external entities or files. Data in files may also flow to processes as input.

### Basic Data Flow Diagram Symbols

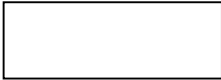
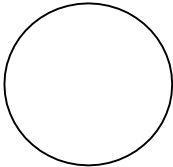
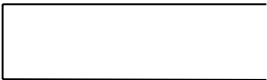

	<p>A source or sink is a person or part of an organization, which enters or receives information from the system, but is considered to be outside the context of data flow model.</p>
	<p>Circles or Ellipse stands for process that converts data in to information. A process represents transformation where incoming data flows are changed into outgoing data flows.</p>
	<p>A data store is a repository of data that is to be stored for use by a one or more process may be as simple as buffer or queue or sophisticated as relational database. They should have clear names. If a process merely uses the content of store and does not alter it, the arrow head goes only from the store to the process. If a process alters the details in the Store then a double-headed arrow issued.</p>
	<p>A data flow is a route, which enables packets of data to travel from one point to another. Data may flow from a source to a process and from data store or process. An arrow line depicts the flow, with arrow head pointing in the direction of the flow.</p>

Table 3.8 Data Flow Diagram Symbols

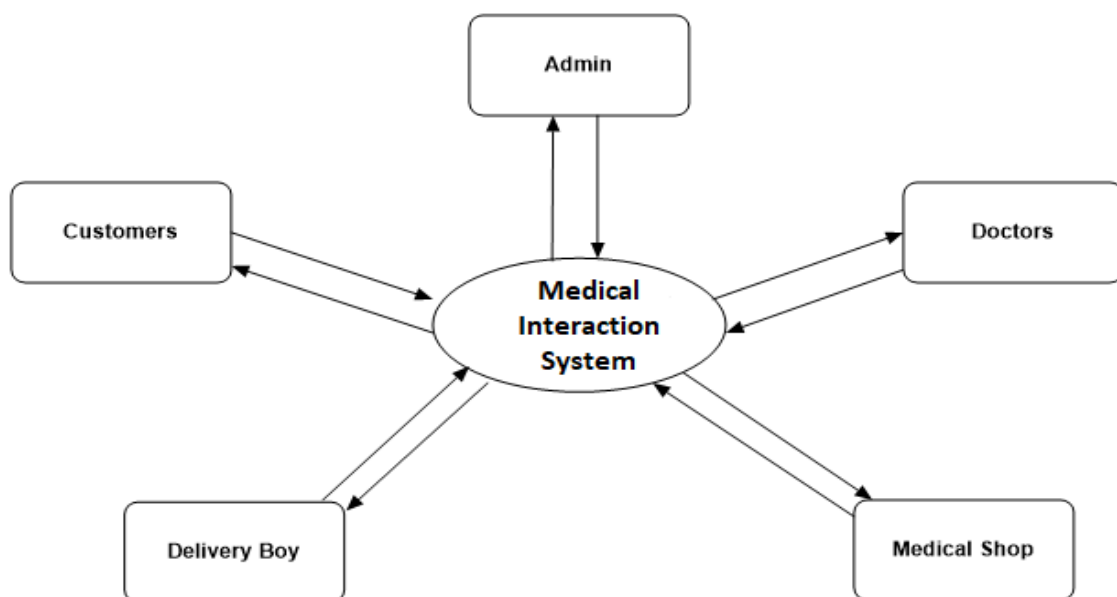
**Rules for constructing a Data Flow Diagram**

1. Arrows should not cross each other
2. Squares, circles and files must bear names
3. Decomposed dataflow squares and circles can have same time
4. Choose meaningful name for dataflow
5. Draw all dataflows around the outside of the diagram

**3.3.3.2 Data Flow Diagram**

Each component in a DFD is labelled with a descriptive name. Process name are further identified with number. Context level DFD is draw first. Then the process is decomposed into several elementary levels and is represented in the order of importance. A DFD describes what data flow (logical) rather than how they are processed, so it doesnot depend on hardware, software, and data structure or file organization.

A DFD methodology is quite effective; especially when the required design

**Level Zero DFD for Medical Interaction System**

**Fig 1 Context level DFD for Medical interaction System**



### Level-1 DFD for Admin

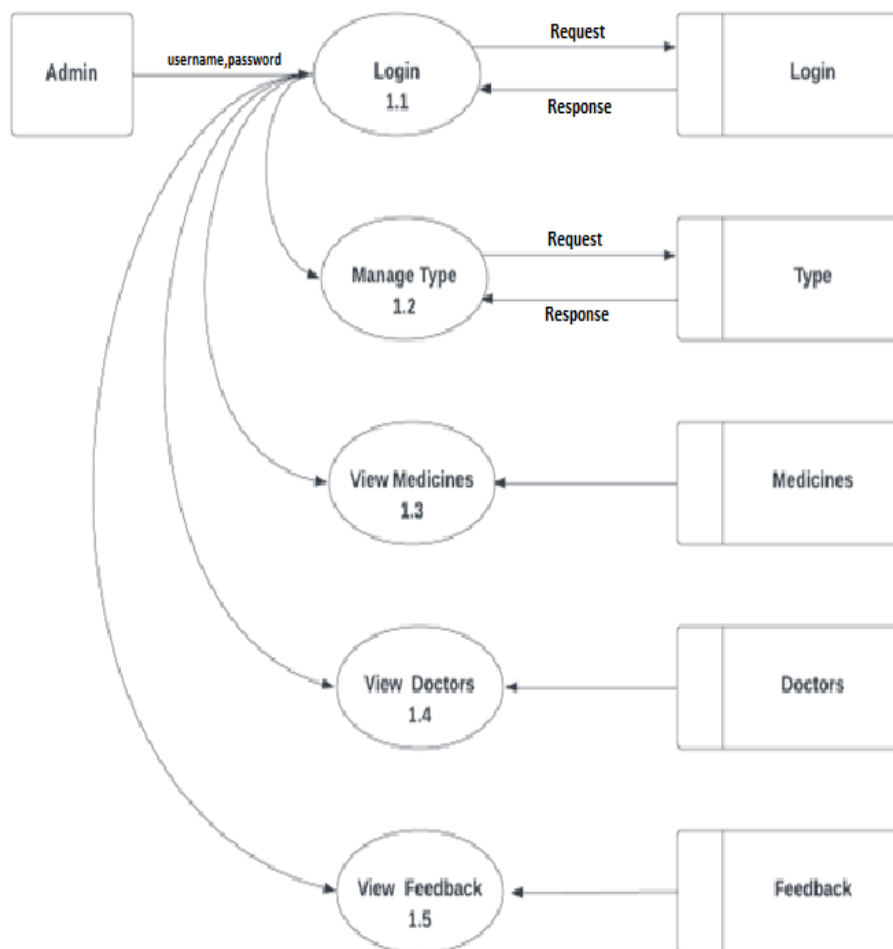
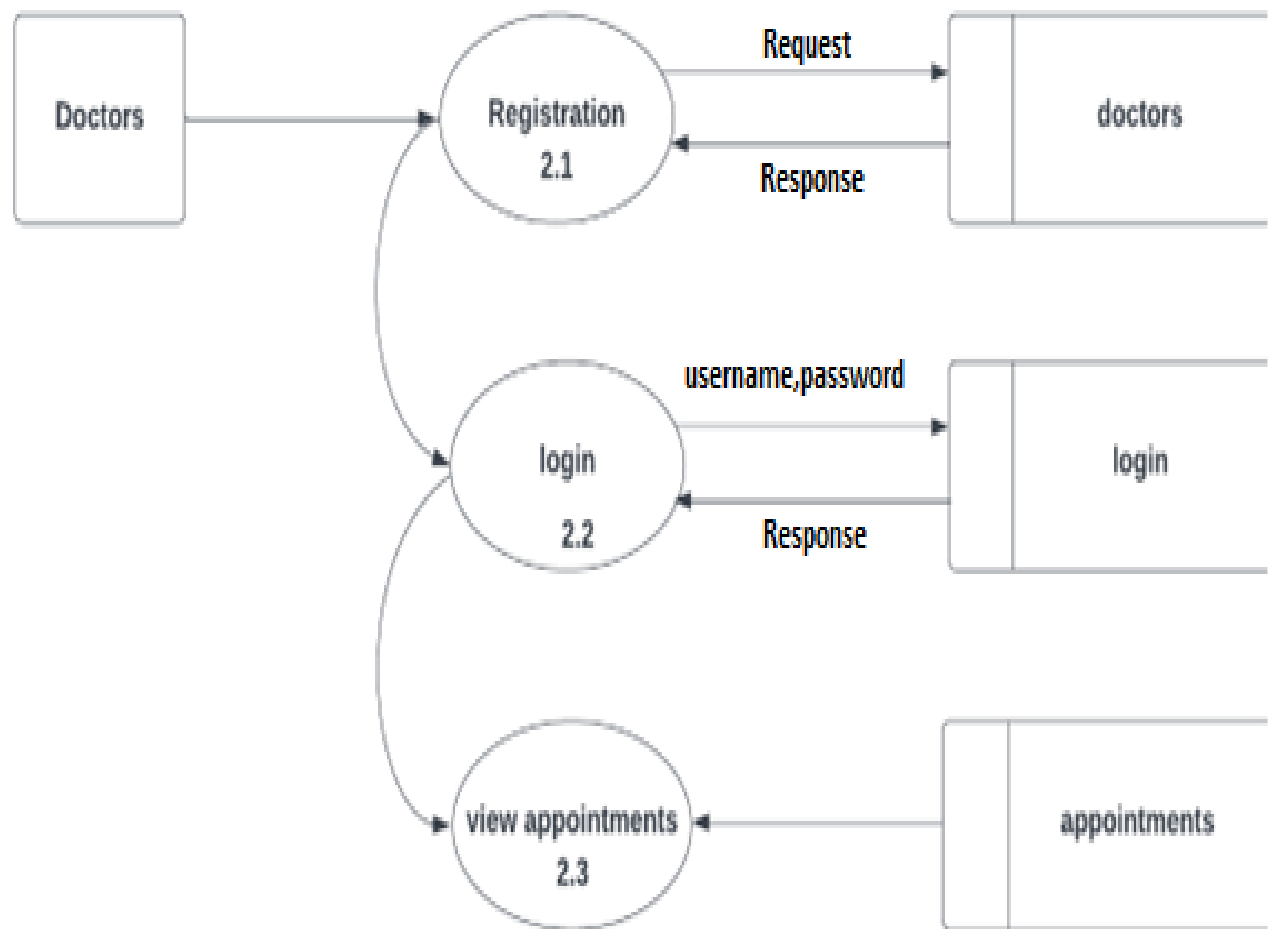
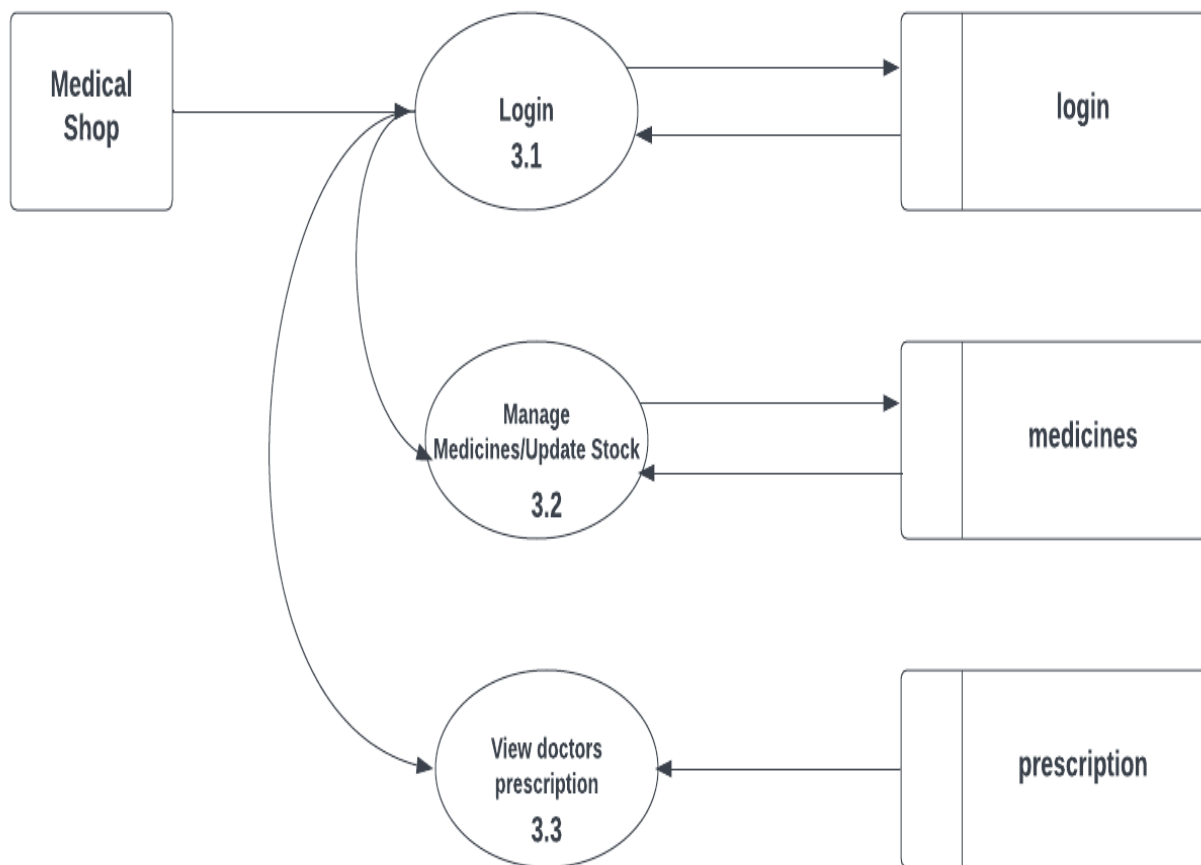
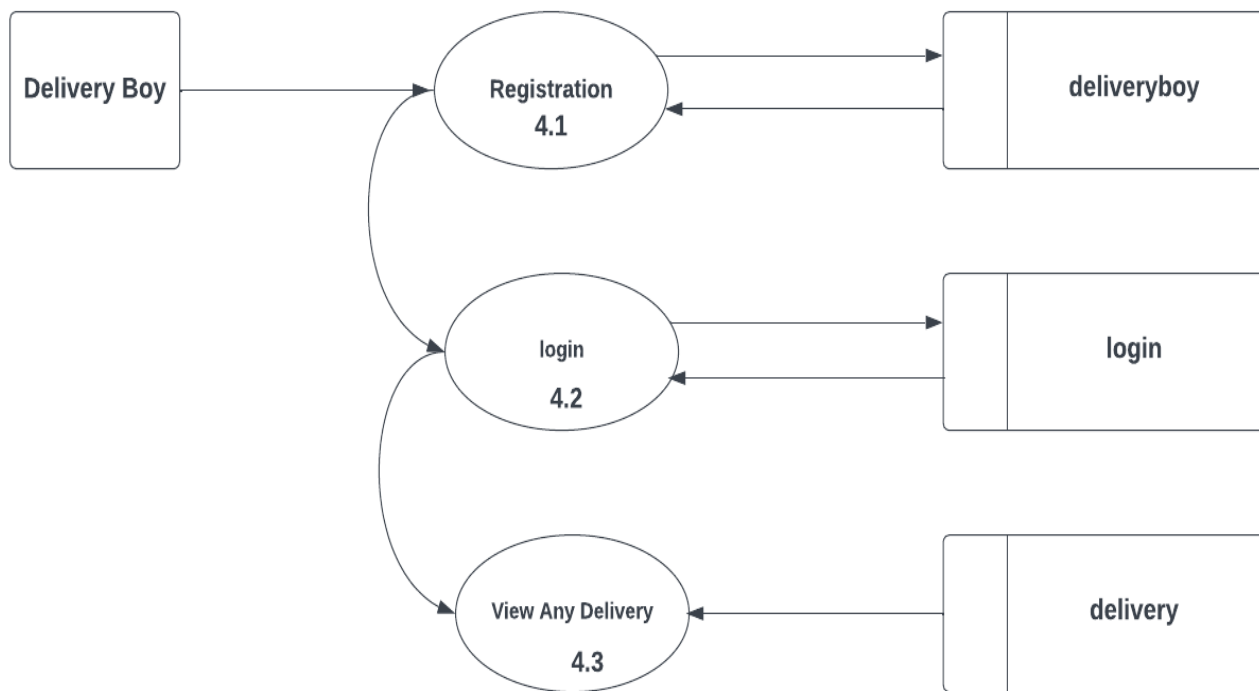
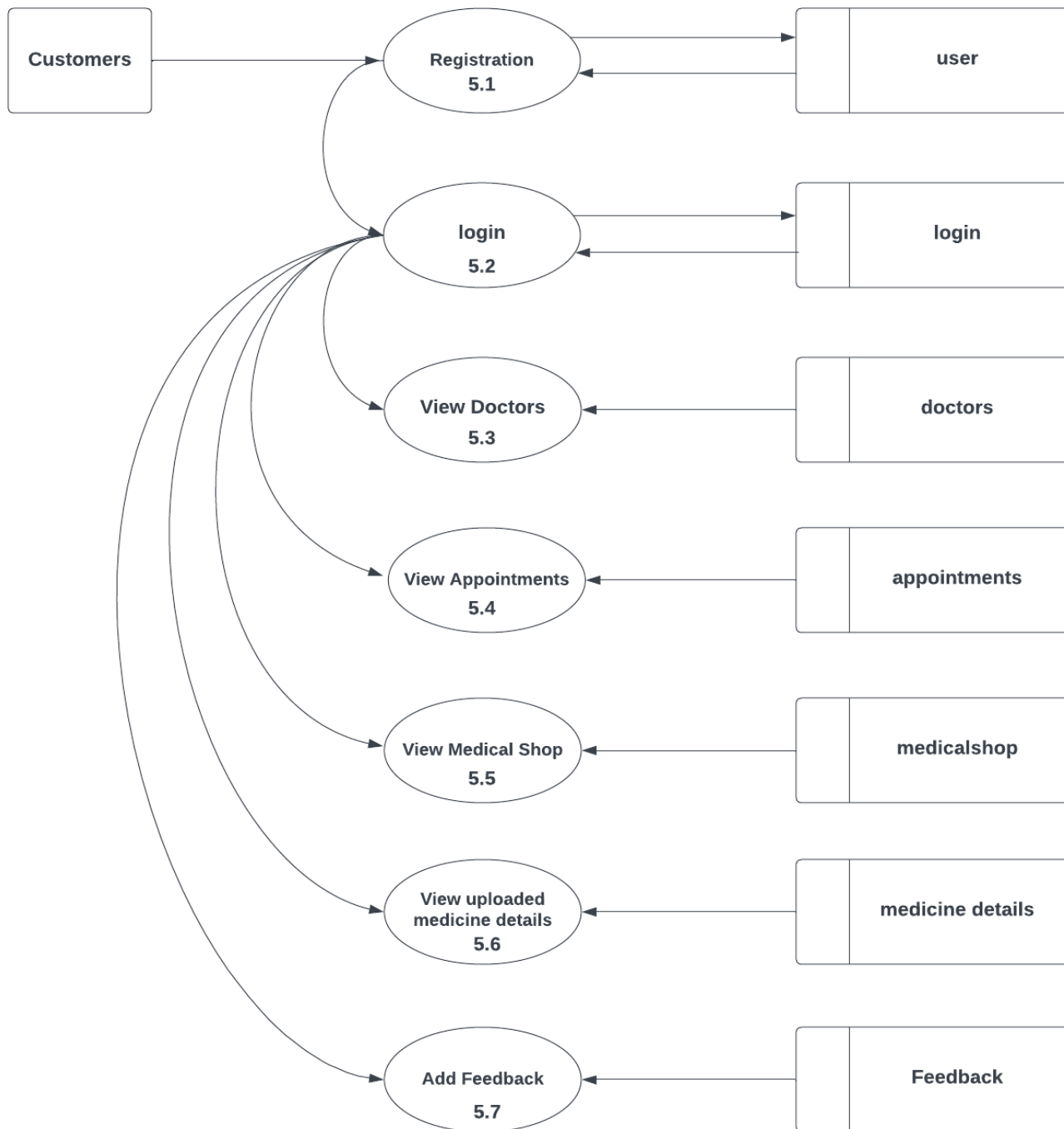


Fig 2. Level-1 DFD for Admin

**Level-1 DFD for Doctor****Fig 3. level-1 DFD for doctor**

**Level-1 DFD for Medical Shop****Fig 4. level-1 DFD for medical shop**

**Level-1 DFD for Delivery Boy****Fig 5. level-1 DFD for Delivery Boy**

**Level-1 DFD for Customers****Fig 6. level-1 DFD for Customers**

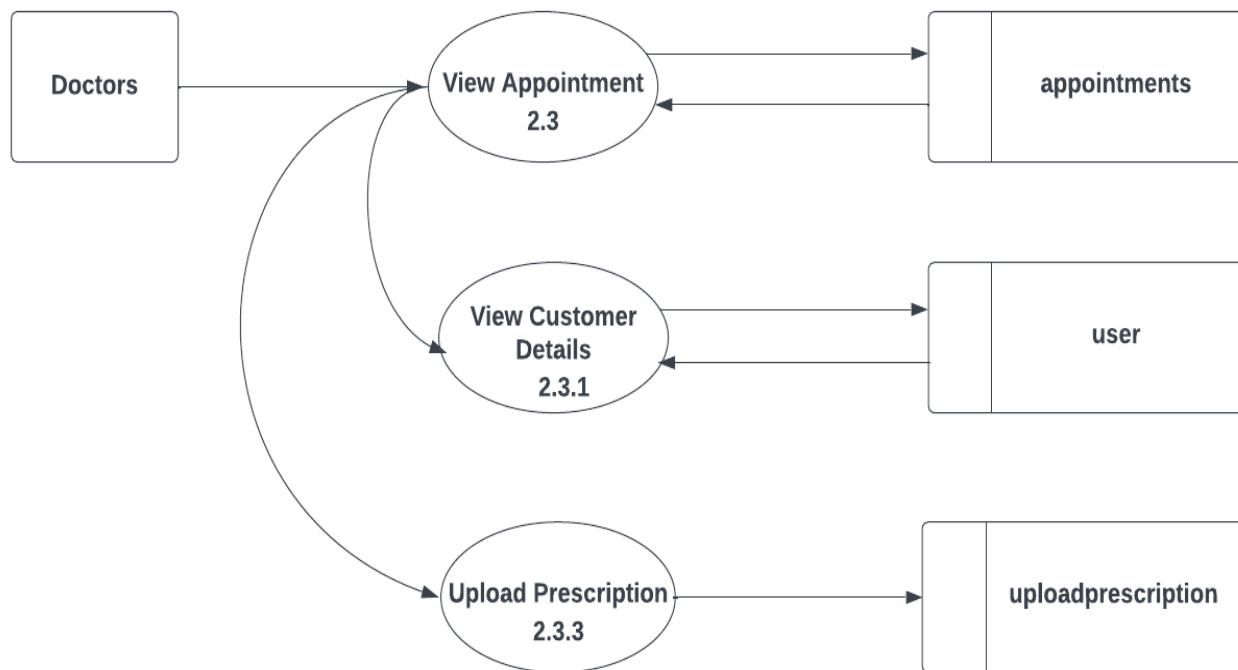
**Level-2 DFD for Doctor**

Fig 7. level-2 DFD for Doctor

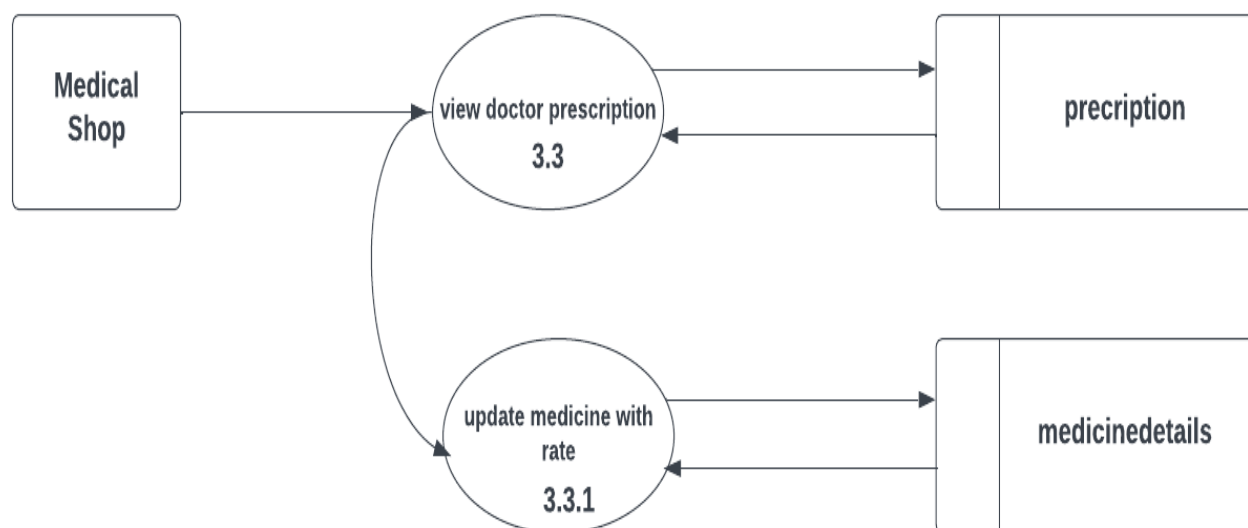
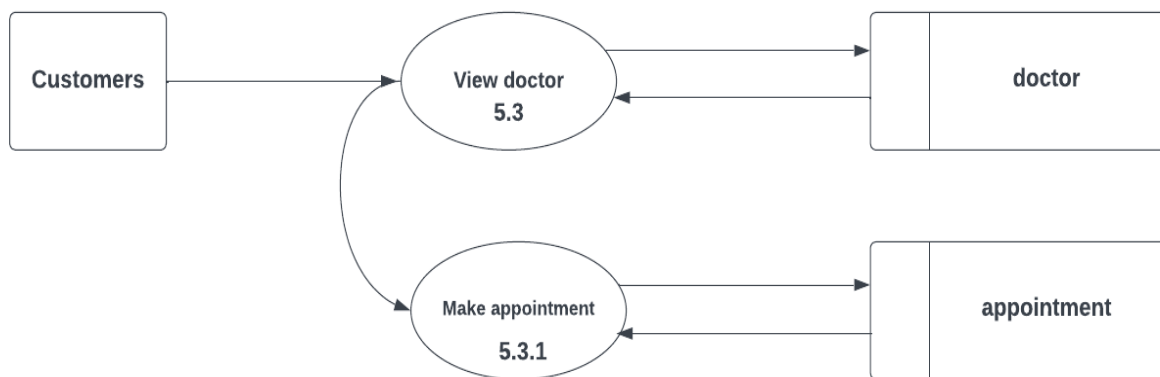
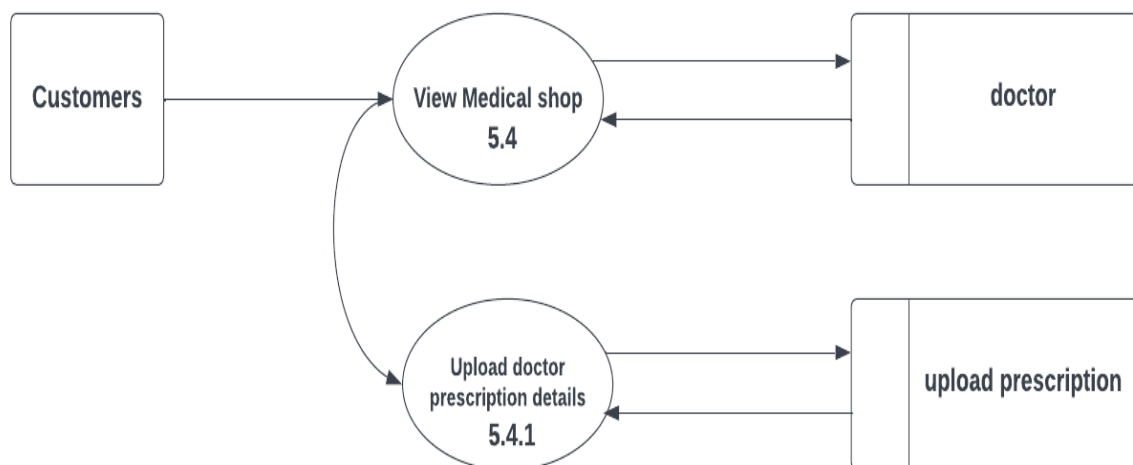
**Level-2 DFD for Medical Shop**

Fig 8. level-2 DFD for Medical Shop

**Level – 2 DFD for Customer-view Doctors****Fig 9. level-2 DFD for Customer view doctors****Level-2 DFD for Customer-view Medical Shop****Fig 10. level-2 DFD for Customer view Medical Shop**

### 3.4 INTERFACE DESIGN

These modules can apply to hardware, software or the interface between a user and a machine. An example of a user interface could include a GUI, a control panel for a nuclear power plant, or even the cockpit of an aircraft. In systems engineering, all the inputs and outputs of a system, subsystem, and its components are listed in an interface control document often as part of the requirements of the engineering project. The development of a user interface is a unique field.

#### 3.4.1 User Interface Screen Design

The user interface design is very important for any application. The interface design describes how the software communicates within itself, to system that interpreted with it and with humans who use it. The input design is the process of converting the user-oriented inputs into the computer-based format. The data is fed into the system using simple inactive forms. The forms have been supplied with messages so that the user can enter data without facing any difficulty. They data is validated wherever it requires in the project. This ensures that only the correct data have been incorporated into system. The goal of designing input data is to make the automation as easy and free from errors as possible. For providing a good input design for the application easy data input and selection features are adopted. The input design requirements such as user friendliness, consistent format and interactive dialogue for giving the right messages and help for the user at right are also considered for development for this project.

Input Design is a part of the overall design. The input methods can be broadly classified into batch and online. Internal controls must be established for monitoring the number of and for ensuring that the data are valid. The basic steps involved in input design are:

- Review input requirements.
- Decide how the input data flow will be implemented.
- Decide the source document.
- Prototype on line input screens.
- Design the input screens.

The quality of the system input determines the quality of the system output. Input specifications describe the manner in which data enter the system for processing. Input design features can ensure the reliability of the system and produce results from accurate data. The input design also determines whether the user can interact efficiently with the system.



### 3. 4.1 Output Design

A quality output is one, which meets the requirements of end user and presents the information clearly. In any system result of processing are communicated to the user and to the other system through outputs. In the output design it is determined how the information is to be displayed for immediate need.

It is the most important and direct source information is to the user. Efficient and intelligent output design improves the system's relationships with the user and helps in decision -making.

The objective of the output design is to convey the information of all the past activities, current status and to emphasis important events. The output generally refers to the results and information that is generated from the system. Outputs from computers are required primarily to communicate the results of processing to the users.

Output also provides a means of storage by copying the results for later reference in consultation. There is a chance that some of the end users will not actually operate the input data or information through workstations but will see the output from the system.

Two phases of the output design are:

1. Output Definition
2. Output Specification

Output Definition takes into account the type of output contents, its frequency and volume, the appropriate output media is determined for output. Once the media is chosen, the detail specification of output documents are carried out. The nature of output required from the proposed system is determined during logical design stage. It takes the outline of the output from the logical design and produces output as specified during the logical design phase.

In a project, when designing the output, the system analyst must accomplish the following:

Determine the information to present. Decide whether to display, print, speak the information and select the output medium.

- Arrange the information in acceptable format.
- Decide how to distribute the output to the intended receipt.

Thus, by following the above specifications, a high-quality output can be generated

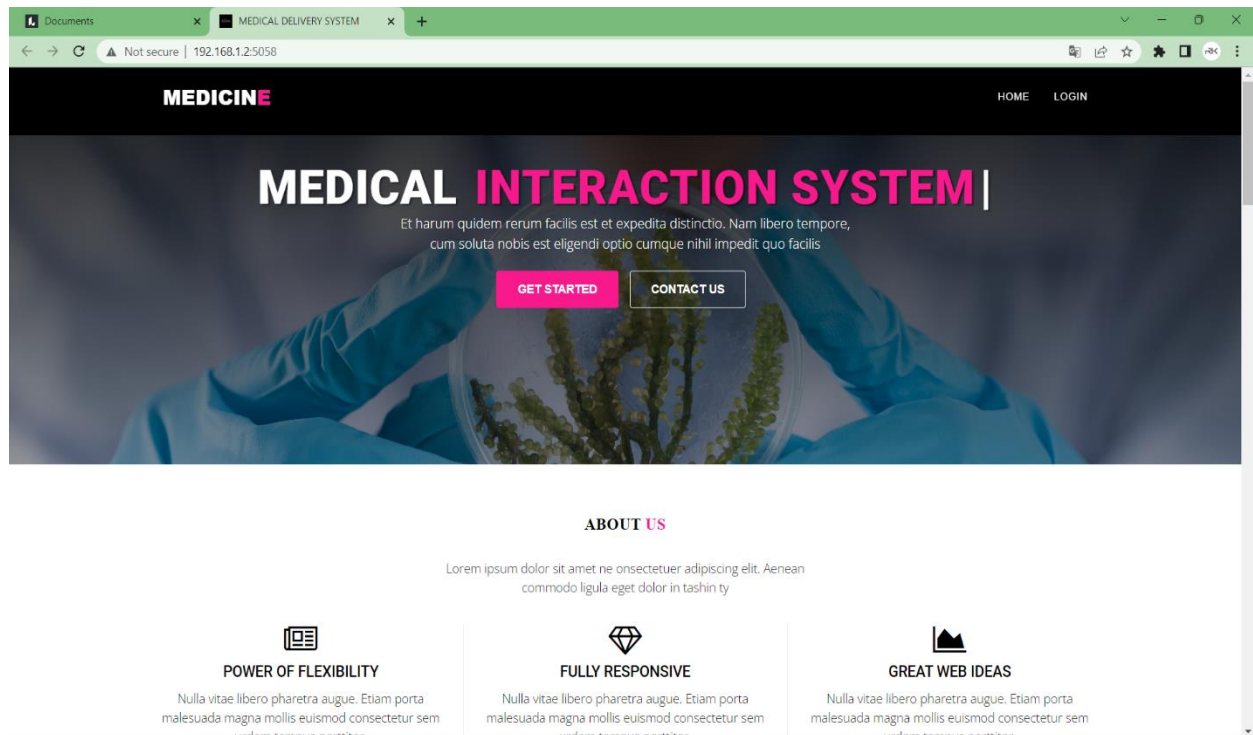


Fig 3.10 Homepage of Medical Interaction System

## 4. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned into a working system. The implementation stage is a systems project in its own right. It includes careful planning, investigation of current system and its constraints on implementation, design of methods to achieve the changeover, training of the staff in the changeover procedure and evaluation change over method.

### 4.1. CODING STANDARDS

Python Flask Framework Coding Standards:

PEP 8: Follow the PEP 8 guidelines for code style and formatting, which includes guidelines for naming conventions, whitespace, line length, and more.

Flask-specific conventions: Follow the Flask-specific conventions for organizing your code, including using the application factory pattern to create your Flask app and separating your app into modules or blueprints.

Routes: Use Flask's routing system to map URLs to functions, and follow the naming conventions for route functions.

Templates: Use Flask's built-in template engine or a third-party library like Jinja2 to render HTML templates, and follow the best practices for organizing and styling your templates.

Error handling: Handle errors and exceptions gracefully, and provide meaningful error messages to users.

Testing: Write automated tests to ensure your Flask app is working as expected, and use a testing framework like pytest or unit test to help you organize your tests.

Android Coding Standards:

Naming conventions: Follow the Android naming conventions for classes, methods, and variables.

XML layouts: Use XML layouts to define your app's user interface, and follow the Android layout guidelines for organizing and styling your views.

Android architecture components: Use Android's architecture components, such as View Model and Live Data, to help manage your app's data and UI.

Error handling: Handle errors and exceptions gracefully, and provide meaningful error messages to users.

Threading: Use threads and handlers to manage background tasks and update your app's UI.

Security: Follow the Android security guidelines to protect user data and prevent unauthorized access to your app.

Testing: Write automated tests to ensure your app is working as expected, and use a testing framework like Espresso or Robolectric to help you organize your tests.

Overall, both Python Flask and Android coding standards emphasize clear, concise, and maintainable code, as well as best practices for error handling and testing. It's important to follow these standards to ensure your code is readable, easy to maintain, and performs well.

## 4.2. SAMPLE CODE

The screenshot displays a web application interface for a medical shop registration. The browser window shows the URL `192.168.1.2:5058/admin/shopreg`. The page header includes the logo 'MEDICINE' and navigation links: HOME, MANAGE, VIEW, and LOGOUT. The main content area features a registration form with the following fields and validation messages:

- NAME:  (Message: Please fill out this field.)
- PLACE:
- CITY:
- DISTRICT:
- PHONE:
- EMAIL:
- USER NAME:  (Message: User Name Must Be Atleast 5 Charecter)
- PASSWORD:  (Message: Password Must Be Atleast 8 Charecter)

Fig 4.2 Sample code

```

from flask import *
from database import *
import uuid
shop=Blueprint('shop',__name__)

@shop.route('/shophome',methods=['get','post'])
def shophome():
    sid=session['sid']
    sname=session['sname']
    return render_template('shophome.html',sname=sname)

@shop.route('/shop_manage_medicine',methods=['get','post'])
def shop_manage_medicine():
    data={}
    sid=session['sid']
    q="select * from type"
    res=select(q)
    data['type']=res
    sid=session['sid']
    if 'submit' in request.form:
        typ=request.form['typ']
        med=request.form['med']
        det=request.form['det']
        rate=request.form['rate']
        m_date=request.form['mdate']
        edate=request.form['edate']

```

```

        qua=request.form['qua']
        q="insert into medicines
values(NULL,'%s','%s','%s','%s','%s','%s','%s')"%(sid,typ,med,det,rate,qua,m_date,edate)
        lid=insert(q)
        flash("ADDED SUCESSFULLY")
        return redirect(url_for('shop.shop_manage_medicine'))
    q="select * from medicines inner join type using(type_id) where
medicalshop_id='%s'"%(sid)
    res=select(q)
    if res:
        data['med']=res
        print(res)
    if 'action' in request.args:
        action=request.args['action']
        id=request.args['id']
    else:
        action=None
    if action=='delete':
        q="delete from medicines where medicine_id='%s'"%(id)
        delete(q)

        return redirect(url_for('shop.shop_manage_medicine'))
    if action=='update':
        q="select * from medicines where medicine_id='%s'"%(id)
        data['updater']=select(q)
    if 'update' in request.form:
        typ=request.form['typ']
        med=request.form['med']
        det=request.form['det']
        rate=request.form['rate']
        qua=request.form['qua']
        q="update medicines set
type_id='%s',medicine='%s',details='%s',rate='%s',quantity='%s' where
medicine_id='%s'"%(typ,med,det,rate,qua,id)
        update(q)
        return redirect(url_for('shop.shop_manage_medicine'))
    return render_template('shop_manage_medicine.html',data=data)

@shop.route('/shop_view_docterp',methods=['get','post'])
def shop_view_docterp():
    data={ }
    sid=session['sid']
    q="SELECT * FROM `uploadprescription` INNER JOIN users USING(user_id) WHERE
medicalshop_id='%s'"%(sid)
    res=select(q)
    print(res)
    data['orders']=res

    return render_template("shop_view_docterp.html",data=data)

@shop.route('/shop_upoload_medi',methods=['get','post'])

```

```
def shop_upload_medi():
    data={}
    prescription_id=request.args['prescription_id']
    data['prescription_id']=prescription_id
    name=request.args['name']
    data['name']=name
    q="SELECT * from medicines"
    res=select(q)
    print(res)
    data['med']=res
    q="select * from uploadprescription where prescription_id='%s'"%(prescription_id)
    res=select(q)
    data['totalamount']=res[0]['totalamount']
    q="select *,medicinedetails.quantity as mdqua from medicinedetails inner join medicines
using(medicine_id) inner join type using(type_id) where prescription_id='%s'"%(prescription_id)
    res=select(q)
    print(res)
    data['mdet']=res
    if 'action' in request.args:
        mid=request.args['id']
        q="select total from medicinedetails where medicine_id='%s' and
prescription_id='%s'"%(mid,prescription_id)
        res=select(q)
        delamt=res[0]['total']
        q="delete from medicinedetails where medicine_id='%s'"%(mid)
        delete(q)
        q="update uploadprescription set totalamount=totalamount-'%s' where
prescription_id='%s'"%(delamt,prescription_id)
        update(q)
        return
    redirect(url_for('shop.shop_upload_medi',prescription_id=prescription_id,name=name))

    if 'submit' in request.form:
        med=request.form['med']
        qua=request.form['qua']
        qua=int(qua)
        q="select * from medicines where medicine_id='%s'"%(med)
        res=select(q)
        avlqua=int(res[0]['quantity'])
        rate=int(res[0]['rate'])
        if avlqua<qua:
            flash("REQUIRED QUANTITY IS NOT AVAILABLE IN OUR STOCK")
        else:
            amt=qua*rate
            print("5555555555555555")
            print(amt)
            q="select * from medicinedetails where prescription_id='%s' and
medicine_id='%s'"%(prescription_id,med)
            print(q)
            res=select(q)
            if res:
                print(res)
                preamt=res[0]['total']
```

```

        q="select * from uploadprescription
prescription_id='%s'"%(prescription_id)
        res=select(q)
        print(res)
        totalamount=res[0]['totalamount']
        newamt=int(totalamount)+int(amt)-(int(preamt))

        q="update uploadprescription set totalamount='%s' where
prescription_id='%s'"%(newamt,prescription_id)
        print(q)
        update(q)
        q="update medicinedetails set total='%s',quantity='%s' where
prescription_id='%s' and medicine_id='%s'"%(amt,qua,prescription_id,med)
        update(q)
        flash("THIS MEDICINE ALREDY ADDED !UPDATED
SUCCEDFULLY")

        return
redirect(url_for('shop.shop_upoload_medi',prescription_id=prescription_id,name=name))
    else:
        q="insert into medicinedetails
values(NULL,'%s','%s','%s','%s','%s')"%(prescription_id,med,qua,rate,amt)
        insert(q)
        q="update uploadprescription set totalamount=totalamount+'%s'
where prescription_id='%s'"%(amt,prescription_id)
        update(q)
        flash("ADDED SUCESSFULLY")
        return
redirect(url_for('shop.shop_upoload_medi',prescription_id=prescription_id,name=name))

    return render_template("shop_upoload_medi.html",data=data)

```

```
@shop.route('/shopviewpro',methods=['get','post'])
```

```
def shopviewpro():
```

```

    data={ }
    prescription_id=request.args['prescription_id']
    data['prescription_id']=prescription_id
    name=request.args['name']
    data['name']=name
    q="SELECT * from medicines"
    res=select(q)
    print(res)
    data['med']=res
    q="select * from uploadprescription"
    res=select(q)
    data['totalamount']=res[0]['totalamount']
    q="select *,medicinedetails.quantity as mdqua from medicinedetails inner join medicines
using(medicine_id) inner join type using(type_id) where prescription_id='%s'"%(prescription_id)
    res=select(q)
    print(res)
    data['mdet']=res

```



```
        return render_template("shopviewpro.html",data=data)
    @shop.route('/shop_assigntodboy',methods=['get','post'])
    def shop_assigntodboy():
        data={ }

        prescription_id=request.args['prescription_id']
        q="select * from deliveryboy"
        res=select(q)
        data['dboy']=res
        if 'submit' in request.form:
            dboy=request.form['dboy']
            q="insert into delivery values(NULL,'%s','%s',curdate())"%(prescription_id,dboy)
            insert(q)

            q="update uploadprescription set status='dispatched' where
prescription_id='%s'"%(prescription_id)
            update(q)
            flash("ASSIGNED SUCCESFULLY")
            return redirect(url_for('shop.shop_view_docterp'))

        return render_template("shop_assigntodboy.html",data=data)
```

## 5. TESTING

Coding conventions are a set of guidelines for a specific programming language that recommend programming style, practices and methods for each aspect of a piece program written in this language. These conventions usually cover file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices, programming principles, programming rules of thumb, architectural best practices, etc. These are guidelines for software structural quality. Software programmers are highly recommended to follow these guidelines to help improve the readability of their source code and make software maintenance easier.

### 5.1 TEST CASES

The objective of system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified and to ensure that the computer system and the associated clerical and other procedures work together. The initial phase of system testing is the responsibility of the analyst who determines what conditions are to be tested, generates test data, produced a schedule of expected results, runs the tests and compares the computer produced results with the expected results with the expected results. The analyst may also be involved in procedures testing. When the analyst is satisfied that the system is working properly, he hands it over to the users for testing. The importance of system testing by the user must be stressed. Ultimately it is the user must verify the system and give the go-ahead.

During testing, the system is used experimentally to ensure that the software does not fail, i.e., that it will run according to its specifications and in the way, users expect it to. Special test data is input for processing (test plan) and the results are examined to locate unexpected results. A limited number of users may also be allowed to use the system so analysts can see whether they try to use it in unexpected ways. It is preferably to find these surprises before the organization implements the system and depends on it. In many organizations, testing is performed by person other than those who write the original programs. Using persons who do not know how certain parts were designed or programmed ensures more complete and unbiased testing and more reliable software.

Parallel running is often regarded as the final phase of system testing. Since he parallel operation of two systems is very demanding in terms of user resources it should be embarked on only if the user is satisfied with the results of testing -- it should not be started if problems are known to exist. Testing is the major quality control measure during software development.

Its basic function is to detect errors in the software. Thus, the goal of testing is to uncover requirement design and coding errors in the program.

Testing is the process of correcting a program with intends of finding an error. Different types of testing are,

1. Unit Testing
2. Integrated Testing
3. Black Box Testing
4. White Box Testing
5. Validation Testing
6. User Acceptance Testing

### **5.1.1 Unit Testing**

In computer programming, unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In this testing we test each module individual and integrated the overall system. Unit testing focuses verification efforts on the smaller unit of software design in the module. This is also known as module testing. The modules of the system are tested separately. The testing is carried out during programming stage itself. In this testing step each module is found to working satisfactory as regard to the expected output from the module. There are some validation checks for verifying the data input given by the user which both the formal and validity of the entered. It is very easy to find error debug the system.

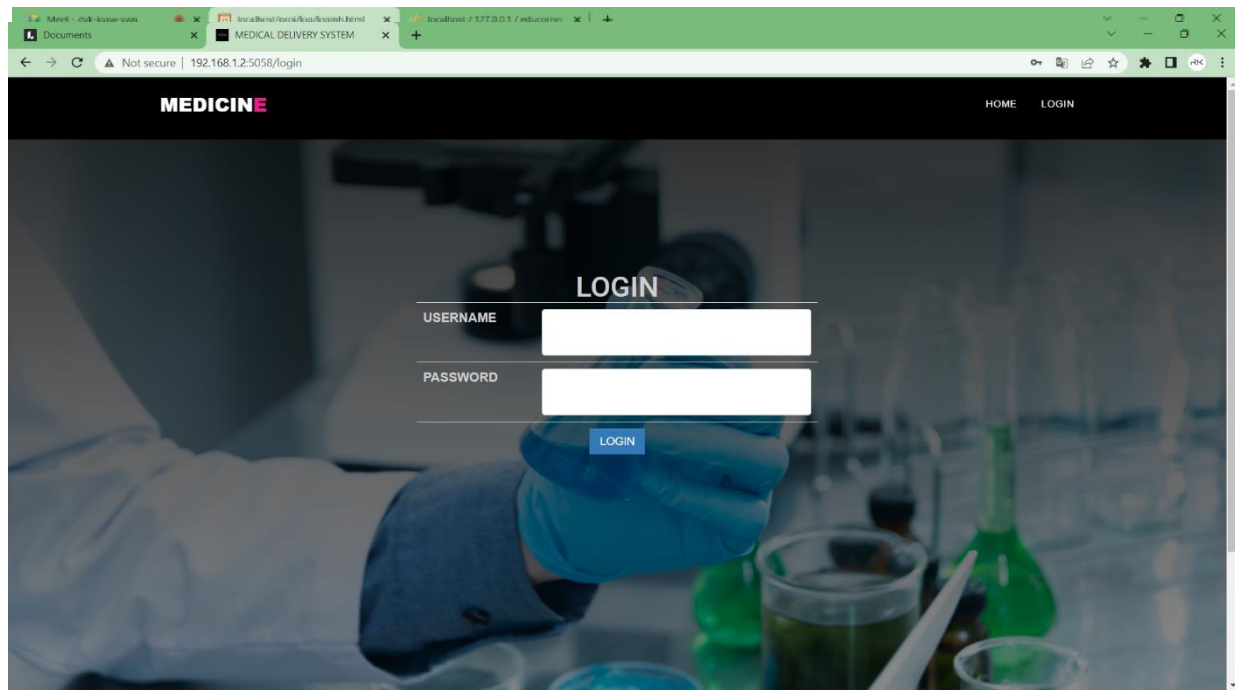


Fig 5.1 Unit testing

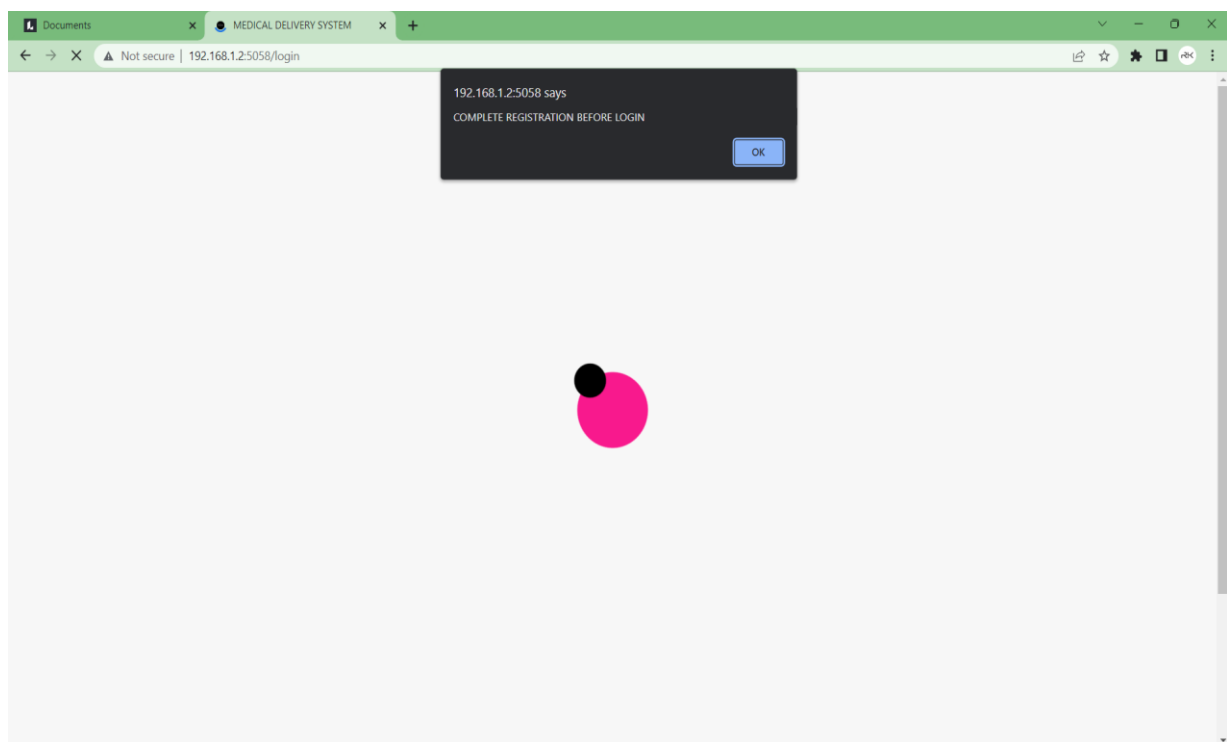


Fig 5.2 Unit testing result

We have continued Unit Testing from the starting of the coding phase itself. Whenever we complete done small sub module, some amount of testing was done based on the requirements to see if the functionality is aligned to the gathered requirements.

### 5.1.2 Integration Testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Data can be lost across an interface; one module can have an adverse effect on the other sub functions when combined by, may not produce the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. This testing was done with sample data. The developed system has run success full for this sample data. The need for integrated test is to find the overall system performance.

Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. Integration testing identifies problems that occur when units are combined. By using a test plan that requires you to test each unit and ensure the viability of each before combining units, you know that any errors discovered when combining units are likely related to the interface between units. This method reduces the number of possibilities to a far simpler level of analysis. Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system. We have performed integration testing whenever we have combined two modules together. When two modules are combined, we have checked whether the functionality works correctly or not through integration testing.

### 5.1.3 Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application (e.g., what the software does) without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher-level testing, but can also dominate unit testing as well. In black box testing the structure of the program is not considered. Test cases are decided solely on the basis of the requirements or the specification of the program or module, and the internals of the module or program are not considered for selection of the test cases.

In the Black Box testing tester only knows the input that can be given to the system and what output the system should give. In other words, the basis of deciding test cases in functional testing is requirements or specifications of the system or module. This form of testing is also called functional or behavioral testing. One advantage of the black box technique is that no

programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested.

### **5.13. White Box Testing**

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

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements. White Box testing is concerned with testing the implementation of the program. The intent of this testing is not to exercise all the different input or output conditions but to exercise the different programming structures and data structures used in the program.

White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement coverage
- Decision coverage

### 5.14. Validation Testing

At the culmination of Black Box testing, software is completely assembled as a package, interface errors have been uncovered and corrected and final series of software tests, Validation tests begins. Validation testing can be defined many ways but a simple definition is that validation succeeds when the software functions in a manner that can be reasonably accepted by the customer. After validation test has been conducted one of the two possible conditions exists.

1. The function or performance characteristics confirm to specification and are accepted.
2. A derivation from specification uncovered and a deficiency list is created

The screenshot shows a web browser window with the URL `192.168.1.2:5058/admin/shopreg`. The page has a header with the logo "MEDICINE" and navigation links: HOME, MANAGE, VIEW, and LOGOUT. The main content area is a registration form with the following fields: PLACE (Muvattupuzha), CITY (Muvattupuzha), DISTRICT (Ernakulam), PHONE (9638527410), EMAIL (thgf), USER NAME (with a validation error message: "Please include an '@' in the email address. 'thgf' is missing an '@'."), and PASSWORD (with a validation error message: "Password Must Be Atleast 8 Charecter"). A blue "REGISTER" button is at the bottom of the form. Below the form, there is a section titled "MEDICAL SHOPS ADDED IN OUR SYSTEM" with a table header: MEDICAL SHOP, PLACE, CITY, DISTRICT, PHONE, EMAIL.

Fig 1. Username validation

This screenshot is identical to the one in Fig 1, showing the same registration form. The email field contains "thgf" and displays the validation error message: "Please include an '@' in the email address. 'thgf' is missing an '@'". The other fields and the overall page layout remain the same.

Fig 2. Email validation

We have given various validations in our forms so that there will be a neat format for the data's that are entered on to the website. We have also given an already existing validation so that the data redundancy is reduced; same data is not entered twice

### **5.15. User Acceptance Testing**

Acceptance Testing is a level of the software testing process where a system is tested for acceptability. User Acceptance testing is the software testing process where system tested for acceptability & validates the end-to-end business flow. Such type of testing executed by client in separate environment & confirms whether system meets the requirements as per requirement specification or not.

UAT is performed after System Testing is done and all or most of the major defects have been fixed. This testing is to be conducted in the final stage of Software Development Life Cycle (SDLC) prior to system being delivered to a live environment. UAT users or end users are concentrating on end-to-end scenarios & typically involves running a suite of tests on the completed system.

User Acceptance testing also known as Customer Acceptance testing (CAT), if the system is being built or developed by an external supplier. The CAT or UAT are the final confirmation from the client before the system is ready for production. The business customers are the primary owners of these UAT tests. These tests are created by business customers and articulated in business domain languages. So ideally it is collaboration between business customers, business analysts, testers and developers. It consists of test suites which involve multiple test cases & each test case contains input data (if required) as well as the expected output. The result of test case is either a pass or fail.

## **5.2 TEST CASE DOCUMENTS**

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application.



## **6. CONCLUSION**

The application works for the benefits of the society and provides an interactive interface between the patient and the doctor. This paper proposed a mobile healthcare application that provides both healthcare providers and patients access to accurate and up-to-date information with less time and effort as well as improved efficiency of the information flow. The main advantage of this application is that doctors will be provided with full history of their patients' health status and patients will hold their data wherever they go. The proposed system will also help Medical Doctors to speed up diagnosis and treatment of patients through the advice and interaction with the patient

## **6.1. FUTURE ENHANCEMENTS**

This application can be improved in the future by adding the following functionalities:

- ✓ Supporting video calls to discuss the problems with doctors.
- ✓ Implement hardware for monitoring status of the patient.
- ✓ Patient's reports can be stored in encrypted form.
- ✓ The system can also be enhanced by using voice recognition feature of the Android.

## **7. APPENDIX**

### **7.1. REFERENCES**

1. [https://en.wikipedia.org/wiki/Third\\_normal\\_form](https://en.wikipedia.org/wiki/Third_normal_form)
2. <https://xampp-windows.en.softonic.com/download>
3. [https://en.wikipedia.org/wiki/Microsoft\\_Word](https://en.wikipedia.org/wiki/Microsoft_Word)
4. <https://www.lucidchart.com/pages/>
5. <http://softwaretestingfundamentals.com/test-case>
6. <http://softwaretestingfundamentals.com/black-box-testing/>
7. <https://www.guru99.com/user-acceptance-testing.html>
8. <https://ecomputernotes.com/software-engineering/feasibilitystudy>

## 7.2. SCREENSHOTS

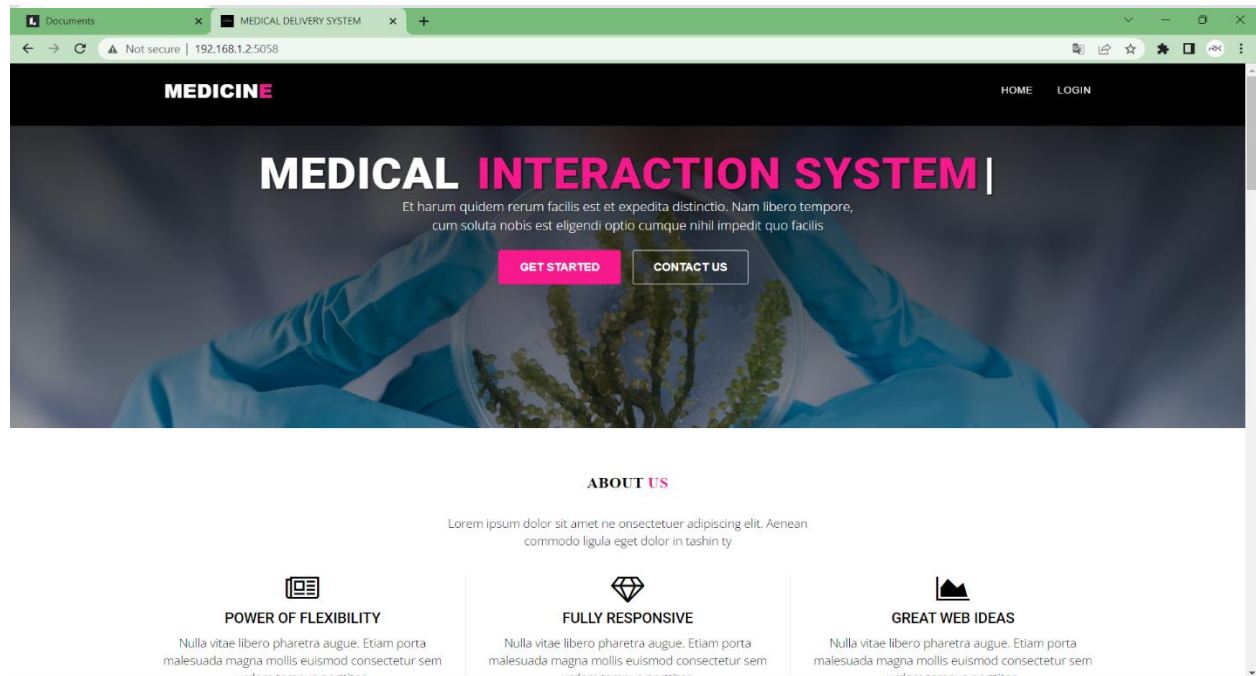


Fig 7.2.1. Home page

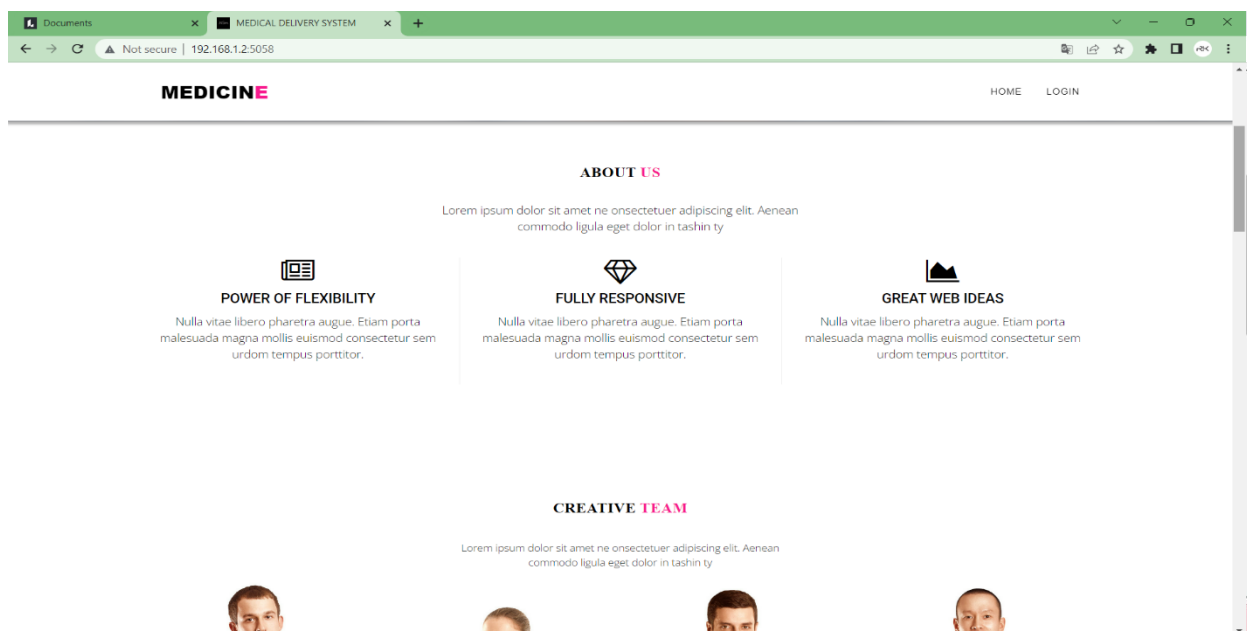


Fig 7.2.2

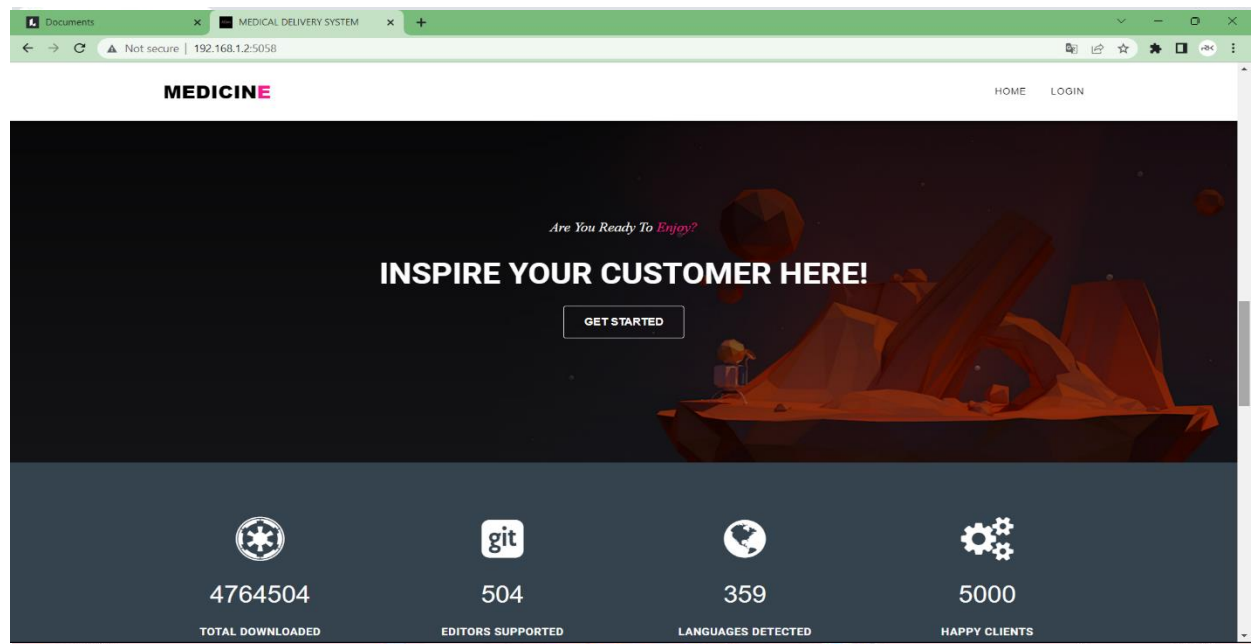


Fig 7.2.3

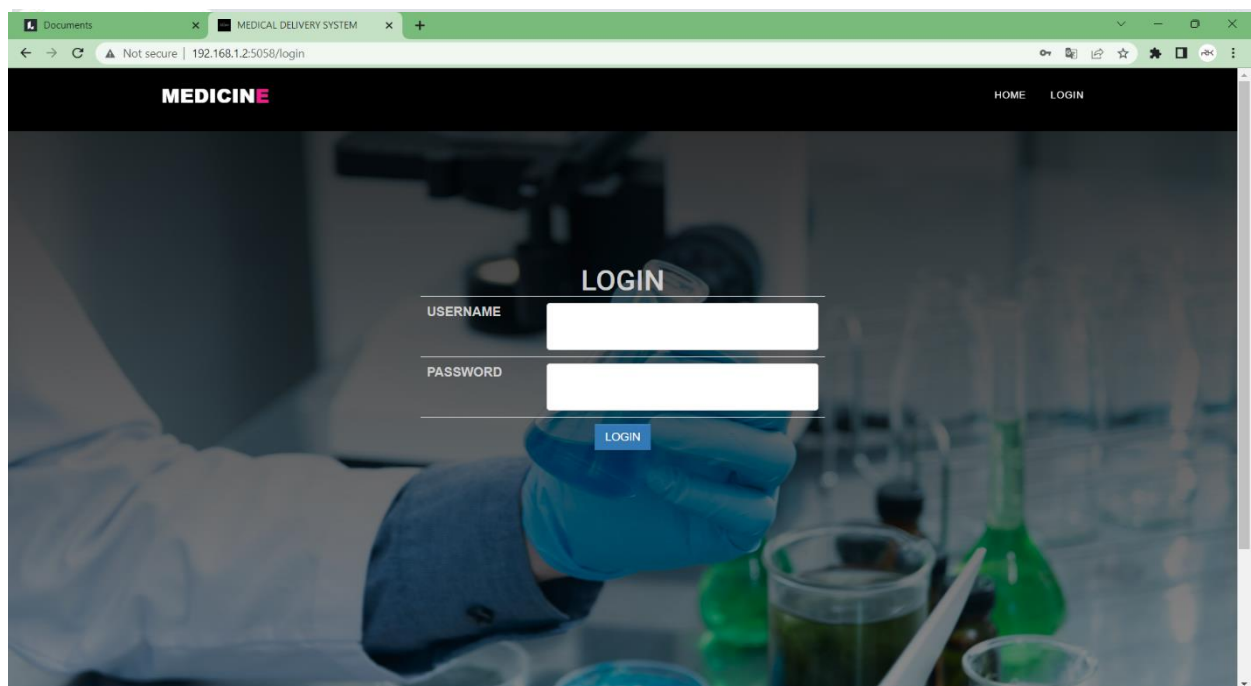


Fig 7.2.4 Login Page

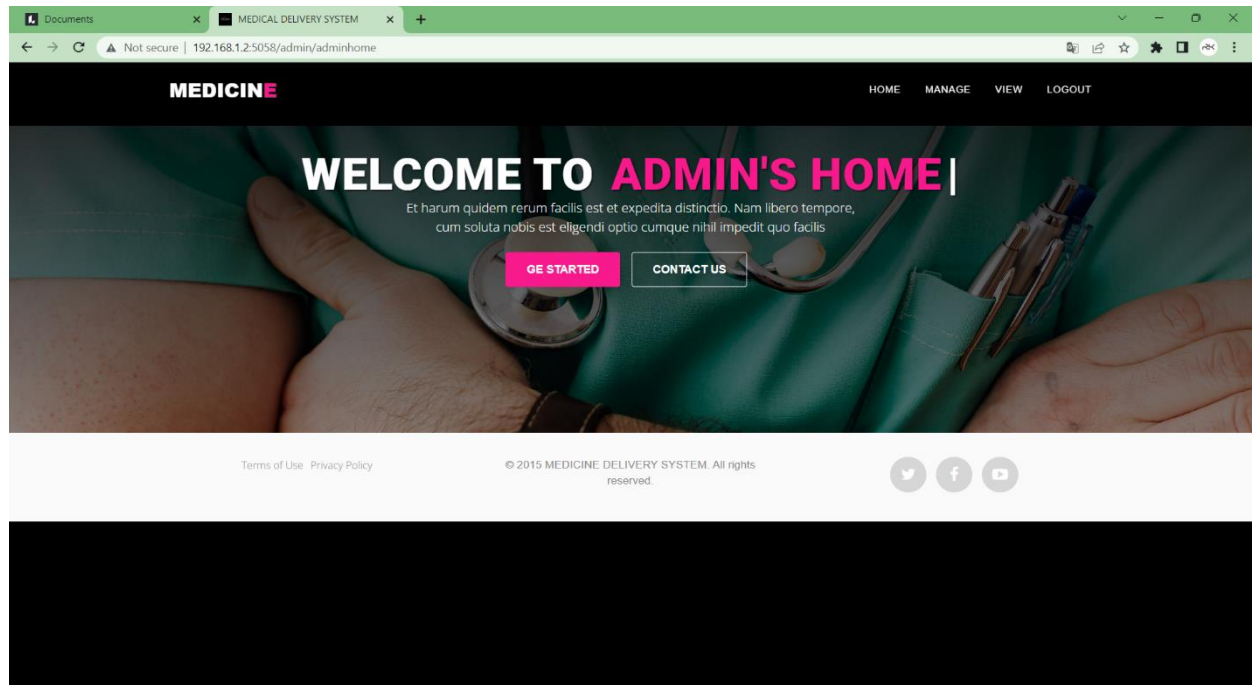


Fig 7.2.5. Admin Page

A screenshot of a web browser displaying the 'MEDICAL SHOP REGISTRATION' page. The browser's address bar shows '192.168.1.2:5058/admin/shopreg'. The page has a dark green header with the 'MEDICINE' logo and navigation links: HOME, MANAGE, VIEW, and LOGOUT. The main content area features a large banner with a background image of a doctor's hands in green scrubs. The banner text reads 'MEDICAL SHOP REGISTRATION' in white. Below the banner is a registration form with the following fields: NAME, PLACE, CITY, DISTRICT, PHONE, EMAIL, USER NAME, and PASSWORD. The USER NAME field has a placeholder text 'User Name Must Be Atleast 5 Charecter' and the PASSWORD field has a placeholder text 'Password Must Be Atleast 8 Charecter'. At the bottom of the form is a blue 'REGISTER' button.

Fig 7.2.7 Medical Shop Registration Page

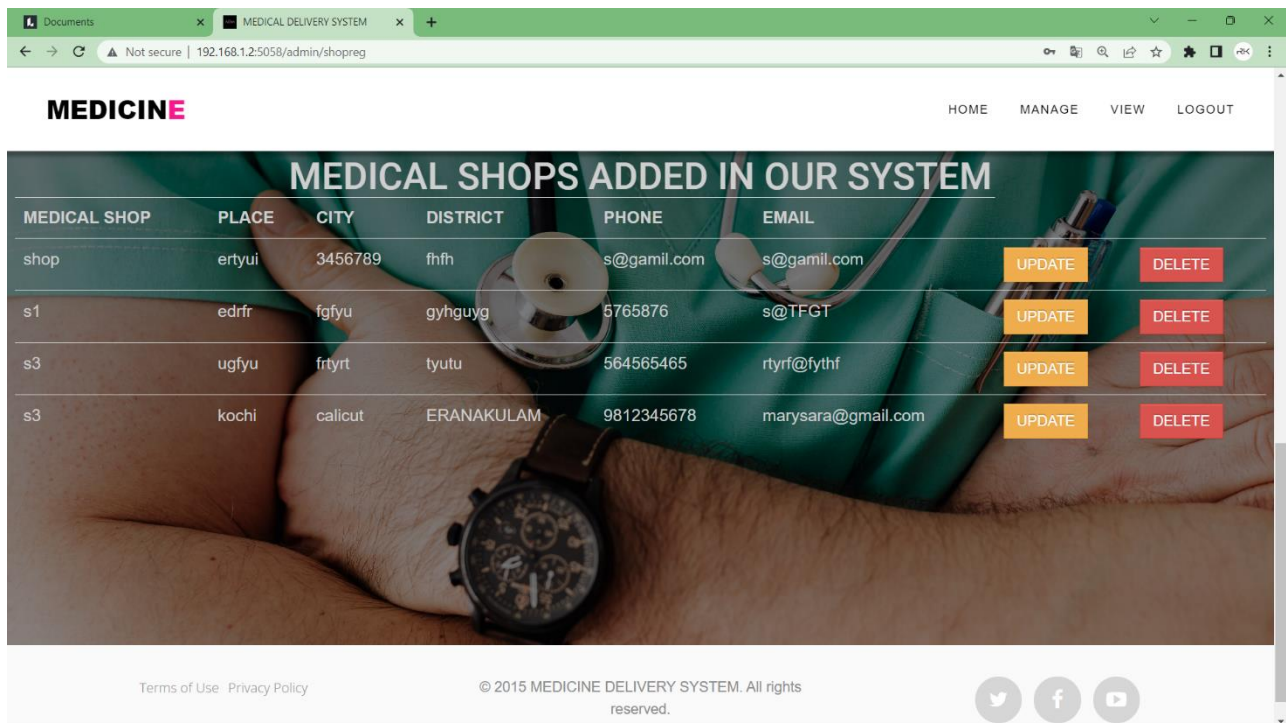


Fig 7.2.8. Admin Manage Medical Shops

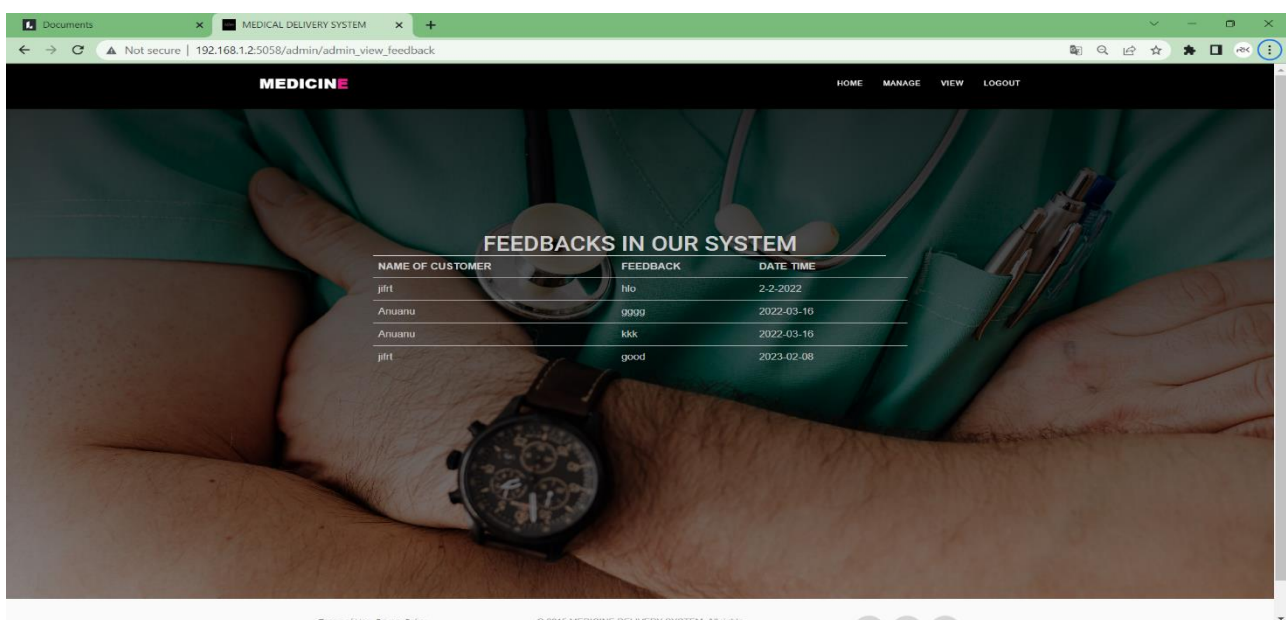


Fig 7.2.9. Admin View Feedback



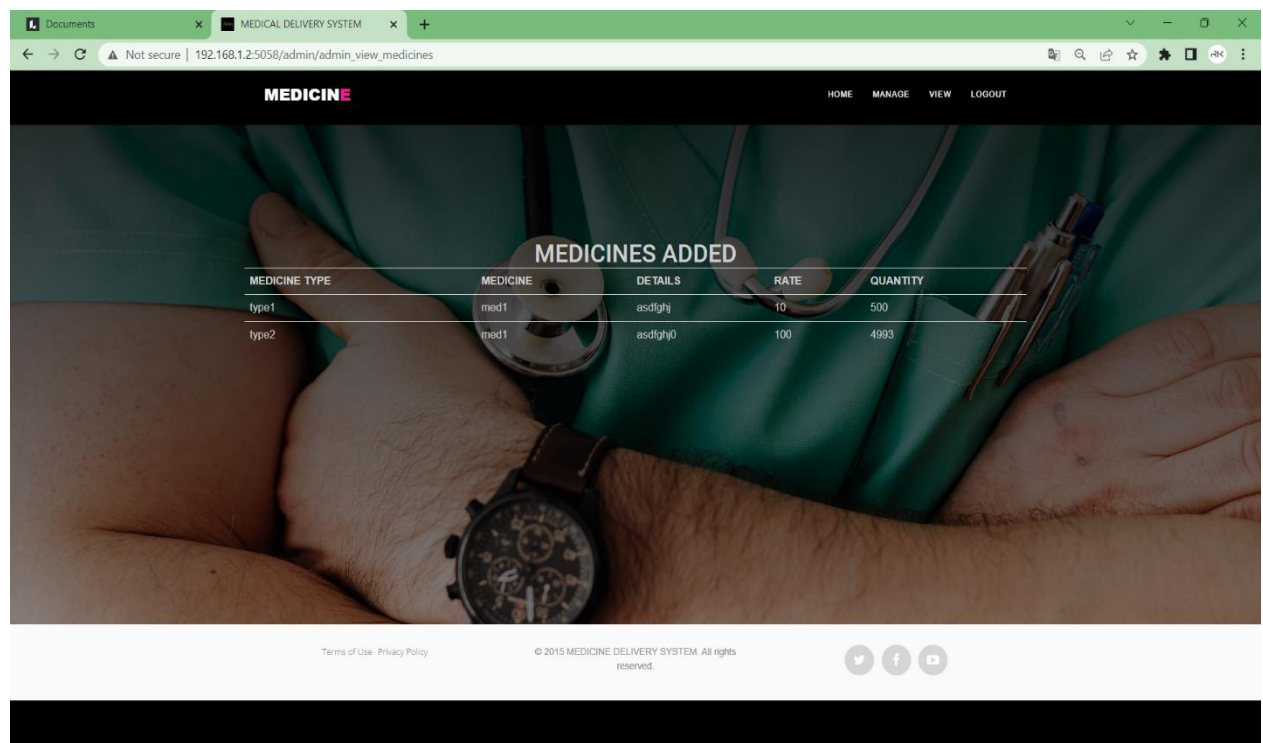


Fig 7.2.10. Admin view Medicines

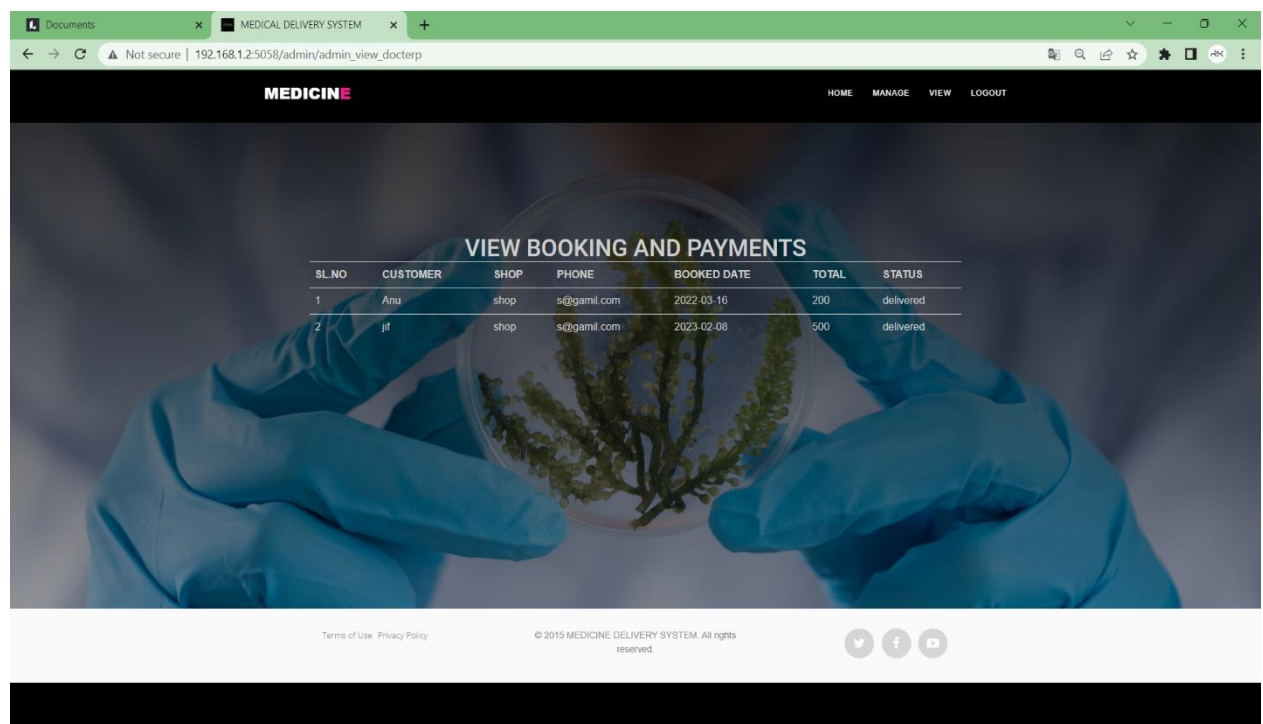


Fig 7.2.11. Admin view bookings and Payments



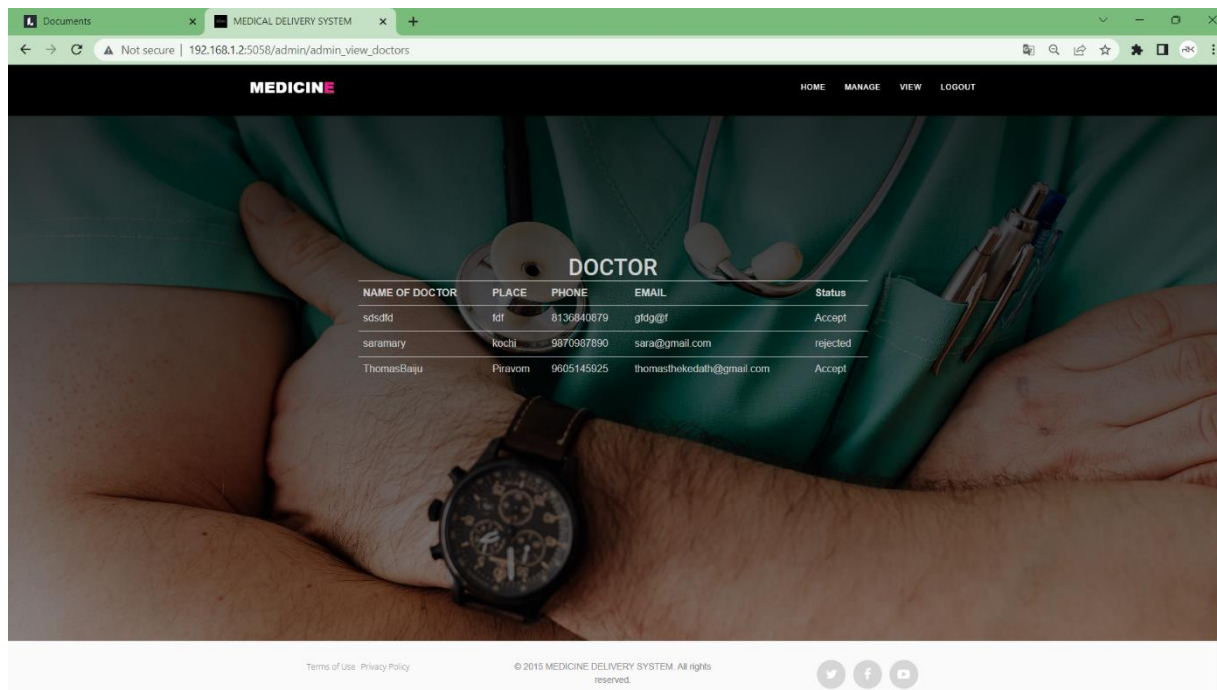


Fig 7.2.12. Admin View Doctors

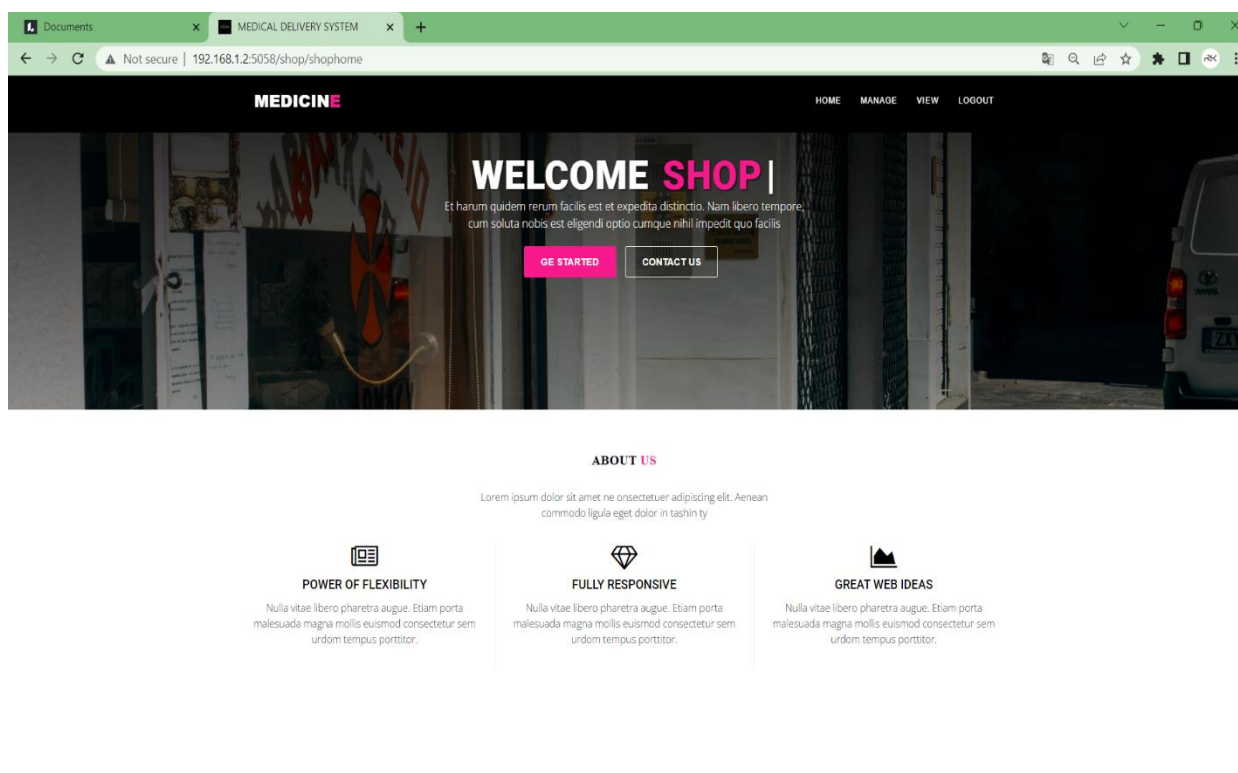


Fig7.2.13. Medical Shop home page

**MANAGE MEDICINES**

MEDICINE TYPE:

MEDICINE:

DETAILS:

RATE:

QUANTITY:

MANUFACTURE DATE:

EXPIRY DATE:

**MEDICINES ADDED**

MEDICINE TYPE	MEDICINE	DETAILS	RATE	QUANTITY
type1	med1	asdfghj	10	500
type2	med1	asdfghj0	100	4993

Fig 7.2.14 Manage Medicines

**VIEW DOCTOR PRESCRIPTION**

SL.NO	CUSTOMER	PHONE	BOOKED DATE	DOCTOR PRESCRIPTION	TOTAL	STATUS
1	jif	8136840879	None		410	pending
2	Anu	8136840879	2022-03-16		200	delivered
3	jif	8136840879	2022-03-16		10	pending
4	jif	8136840879	2023-02-08		500	delivered
5	jif	8136840879	2023-02-10		0	pending

Fig7.2.15 View Doctor Prescriptions