**1. INTRODUCTION**

The idea on which Find Medi is built is to have a means for buying medicines online from trusted pharmaceutical companies or medical stores and also for great maintenance, update of the records of both customers and medical shops through an integrated system. This system reduces time consumption and human effort in maintaining the medical records of medicines and customers easily at any time. In this system all the apprehensive fields like purchasing and selling medicines, maintaining their inventory, generating sales invoices and generating reminders of expiry date about medicines and the details of the customers are done effortlessly without duplication of data.

* 1. **Motivation**

The main motivation of our project is one of the problems which we all faced at some point of time. Many times as customers we see ourselves looking for some medicines and find only some of them and we go looking for them in other shops. Also, many of the local medical retailers may not have the proper system for keeping all the records of sales and medicines in stock and mostly paper based. So, with the current technologies why cant we have a simple yet efficient medical management system for both customer and dealer

* 1. **Problem Definition**

The maintenance of medical records is a hefty work but needed compulsorily. When there are increasing numbers of medicines in the market for a dealer to sell he finds it very difficult to keep track of all the medicines by their names, quantity available, expired medicines in stock, medicines that are out of stock, customer details, sales tracking etc., Also, a customer needs to find his/her medicines by searching many shops sometimes.

* 1. **Objective of Project**

The main objective of the project is to have a means for buying medicines online from trusted pharmaceutical companies or medical stores and also for great maintenance, update of the records of both customers and medical shops through an integrated system. This system reduces time consumption and human effort in maintaining the medical records of medicines and customers easily at any time.

* 1. **Limitations of the project**
* A Customer can buy the required medicines or can access only his/her own medical records through his own unique account but not others.
* Customers and Dealers need to be registered before they use the system.
* The Admin itself takes cares of adding only trusted dealers.
* Not everyone can become a dealer only authorized ones can access the system and the authorization is given by the admin itself.
* Customers must provide some necessary details in order to getting registered.
* The system doesn’t provide the suggestions for which medicine to be used for which disease.

1. **LITERATURE SURVEY**

**2.1 Introduction**

A literature survey is both summary and explanation of the complete and current state of knowledge on a limited topic as found in academic books and journal articles. There are two kinds of literature review you might right at universities: one that students are asked to process in their field, and other that is written as a part of an introduction to, or preparation for, a longer work, usually a thesis or project report. The focus and perspective of your review and the kind of hypothesis or thesis argument you will be determined by what kind of review you are writing. One way to understand the difference between these two types is to read published literature review or the first chapter of these and dissertations in your own subject area. Analyze the structure of their argument and the way they address the issues.

**2.2 Existing System**

In the existing system all the details of medicines and customers are taken manually. All the details of the stock calculations and transactions are recorded manually which required more paper work. Customers need to make much effort by physically going and searching them at different shops. Searching the information about medicines or customers is complex.

**2.3 Disadvantages of Existing System**

* Record keeping will be very difficult with the existing system.
* Problems like data loss or data duplication may occur.
* Customer must visit the shop wasting time and energy.
* Removal of expired medicines are frequently forgotten.

**2.4 Proposed System**

The proposed system is a user friendly system which requires less time and energy. All the information can be stored in the database where we can make any changes to the data like modification or deletions easily without any data loss age or data duplication. In this system report generation is very fast and uncomplicated. All the calculations and transaction details are done within no time. Searching for any medicine and details of the customer is easy. This system provides security to the data.

The customer can buy is medicines directly from the pharmaceutical companies or dealers through his own account given and also keep his or her previous medical records and view them whenever they could.

The Dealer can have the complete information needed to run his shop from having the previous sales records to customer details,from records of expired medicines in the stock to medicines out of stock.Quick update for the quantity of medicines left will be made after each purchase.

The Admin will be the one to authorize authenticated and trusted dealers and new customers and dealers authentications are managed by himself.

1. **ANALYSIS**

**3.1 Introduction**

This segment of the document may consist of software requirement specifications, flowcharts and flow diagrams describing the project.

**3.2 Software Requirement Specification**

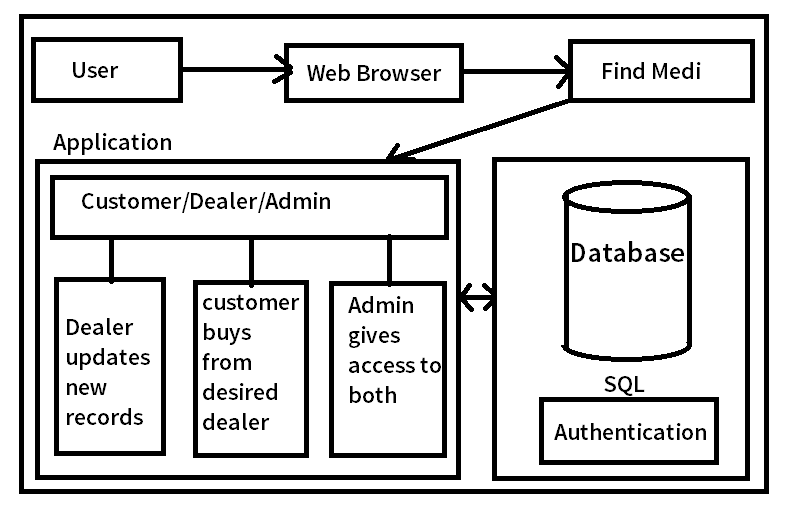
**Minimum Software Requirements:**

* Operating system : Windows 7
* Tools : Netbeans 7.2.1
* Coding Language : Java/J2EE
* Database Management : MySql
* Technologies used : JSP, Java Script, HTML.

**Minimum Hardware Requirements:**

* Processor : Pentium Dual Core
* RAM : 1 GB RAM
* Monitor : 15” LED
* Hard Disk : 120 GB
* Keyboard : Standard 104 Keys
* Mouse : 3 Buttons

**3.3 Architecture**



**Fig 3.1 FIND MEDI Architecture**

In the above architecture we see that the users are accessing the application using a web browser. From there we see that they use their login credentials and if the user is a dealer he uploads the records which then will be stored in the database. The customer can buy the medicines and new quantities will be updated into the database. The admin gives access to both the customer and dealer.

**3.4 Flow Chart**

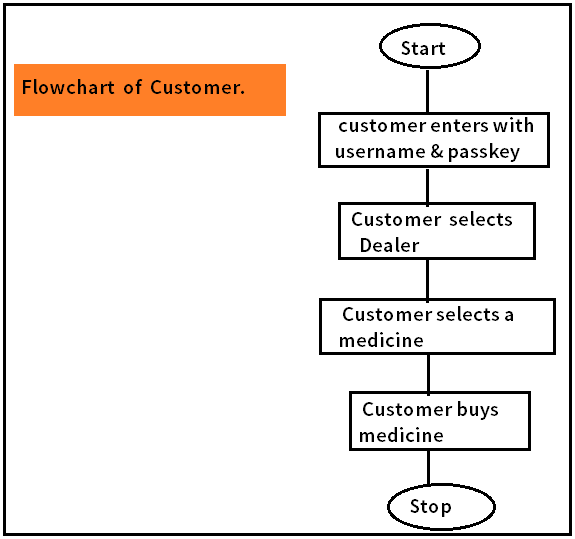


Fig 3.2 Flowchart for Customer

In the above Flow Chart we see that the customer enters the user-id and password then he selects a dealer and medicine which he wishes to buy and then he gets a list of medicine companies from which he selects a specific medicine and buys them and logout.

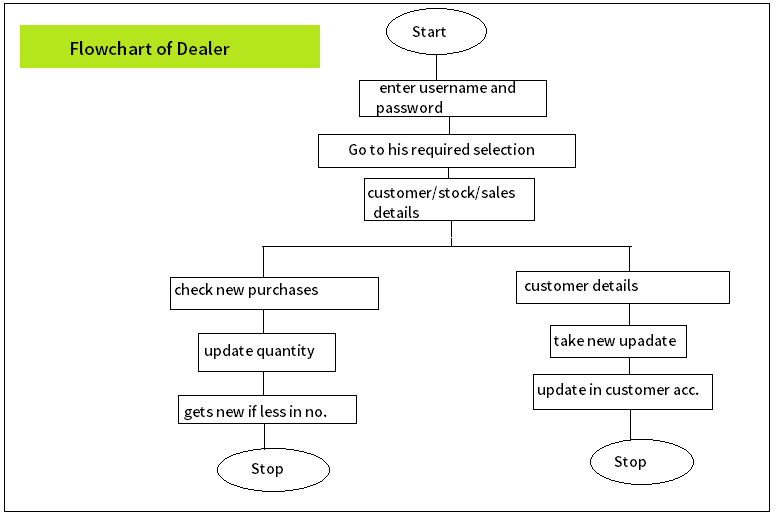


Fig 3.3 Flowchart for Dealer

In the above flowchart for the Dealer we see that there are two ways one in which they select customer details from menu then select their purchase record followed by the medicine that they want to buy and then update new stock. In the second option we see that they select new purchases from the top menu and display the list of medicines buyed by them.

**4. DESIGN**

**4.1 Introduction**

Software design is the process of implementing software solution to one or more sets of problems. This section consists of data flow diagram, module design and organization of the project.

**4.2 Entity Relationship Diagram**

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

An ER model is typically implemented as a [database](https://en.wikipedia.org/wiki/Database). In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a [relational database](https://en.wikipedia.org/wiki/Relational_database) a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

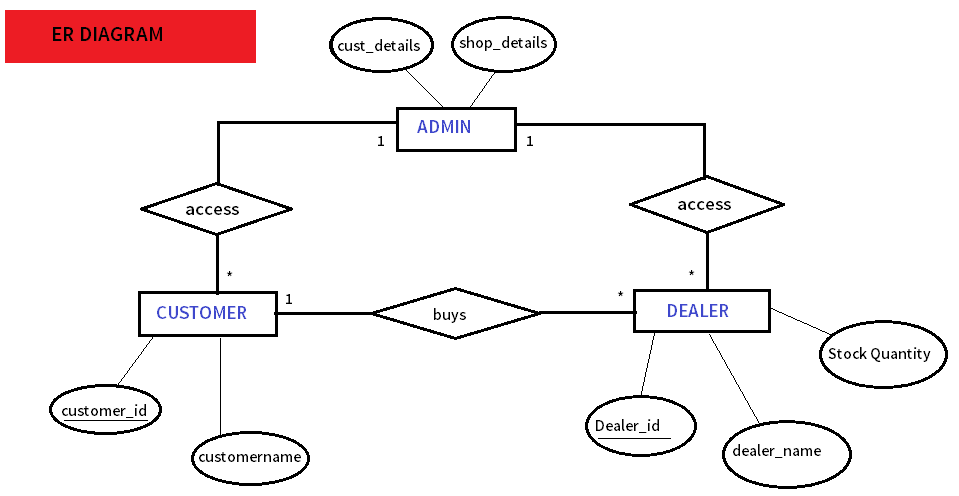


Fig 4.1 Entity Relationship Diagram

* 1. **UML DIGRAMS**

**4.3.1 Use Case Diagram**

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

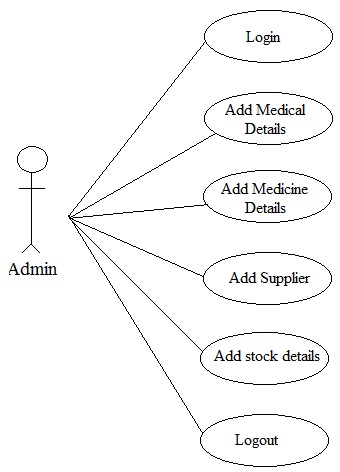
****

Fig 4.2 Use Case diagram

**4.3.2 Class Diagram**

Class diagram is a static diagram. It represents the static view of an a system. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

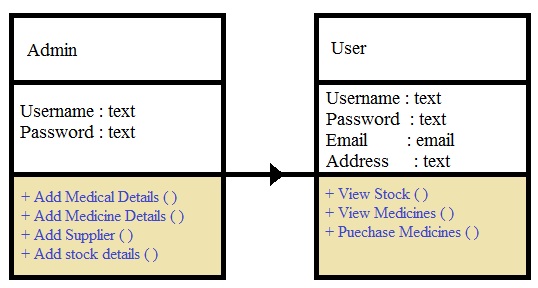
****

Fig 4.3 Class diagram

**4.3.3 Sequence Diagram**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple run time scenarios in a graphical manner.

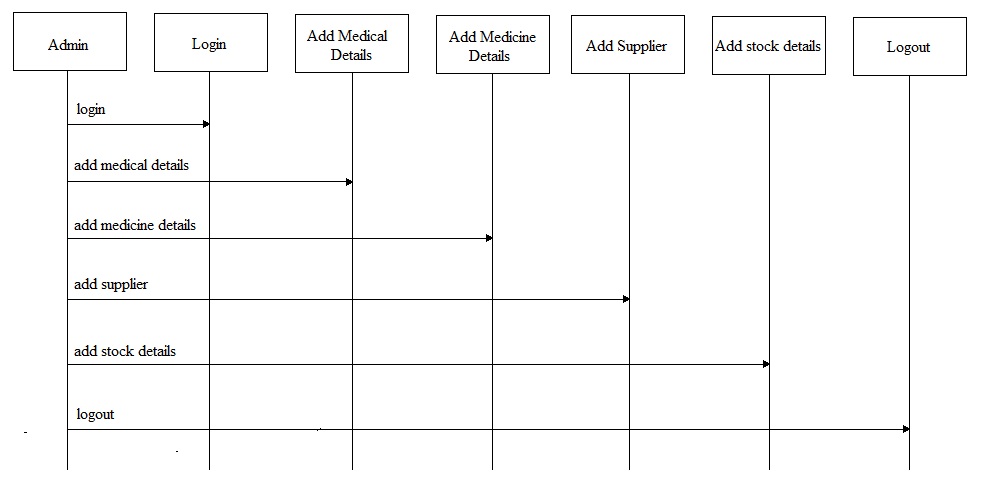
****

Fig 4.4 Sequence diagram for Admin.

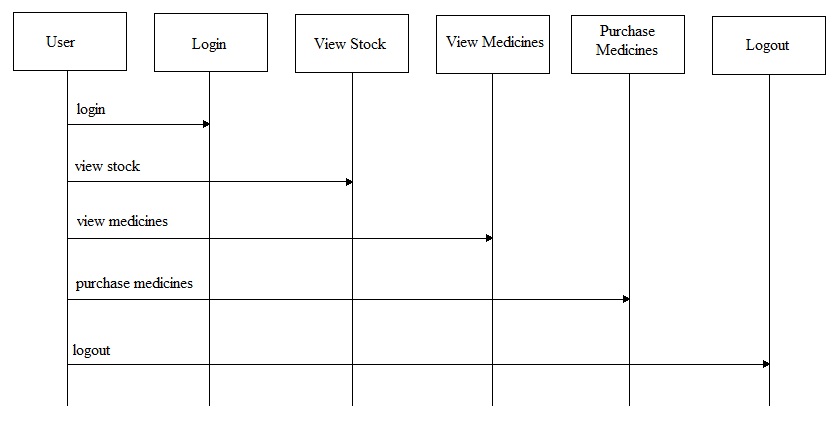
****

Fig 4.5 Sequence diagram for User

**4.3.4 Activity Diagram**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

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

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

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

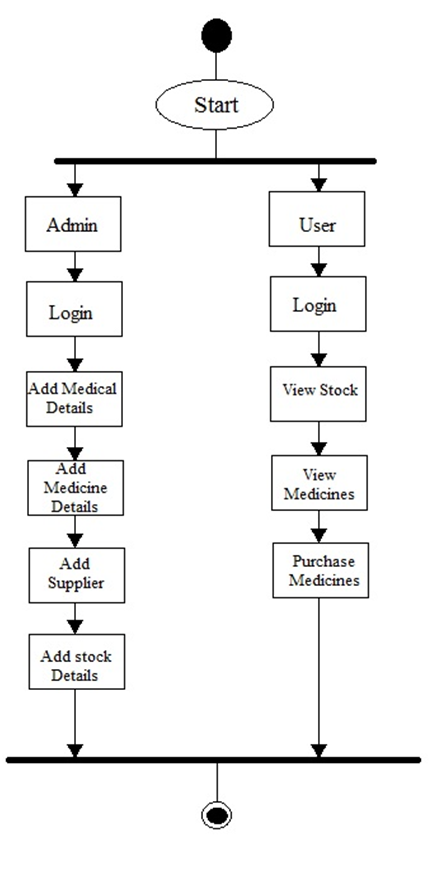


Fig 4.6 Activity diagram for Admin and Users

**4.4 Modules**

There are many modules involved in the process of both the customer and Dealer purchase and sell work flow along with the Admin request processing and Authorization access.

**4.4.1 Admin Module**

Admin will monitor all the system. Admin will get login by entering the username and password. Admin can view all the information entered by the user. Admin will add, delete, update and modify all the details of medical record, medicine details, stock details and supplier details.

**4.4.2 User module**

User either customer or Dealer need to get registered in this system by giving all the information. User can get login by entering the unique username and password. User can view stock details, view medicine details and purchase medicines.

1. **IMPLEMANTATION**

**5.1 Introduction**

In project implementation or project execution, we put it all together. Project planning is complete, as detailed as possible, yet providing enough flexibility for necessary changes. In a customer-contractor relationship, the contract is signed, based on the right decisions about the contract structures, and including clauses for change and claim management.

Now we apply all the tools we prepared in order to keep ourselves in control of the project. As project managers and sub-project managers we have to make sure that we, together with all our team members, take action, in-line with the plan and / or contract record and document all the work, work results, special events, decisions about changes, implementation of changes, etc. analyze, communicate, report, and document status and results of action, in-line with the plan and / or contract take decision if and what kind of change we need, in case any result (or action) is not as required implement agreed changes, in-line with the plan and/or contract.

In project implementation, we manage implementation of all our project plans, following the triple constraint:

1) Project Scope Management

2) Project Time Management

3) Project Cost Management

**5.2 Technologies**

**5.2.1 Netbeans 7.2.1**

**NetBeans** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). NetBeans allows applications to be developed from a set of modular [software components](https://en.wikipedia.org/wiki/Software_component) called modules. NetBeans runs on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [macOS](https://en.wikipedia.org/wiki/MacOS), [Linux](https://en.wikipedia.org/wiki/Linux) and [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)). In addition to Java development, it has extensions for other languages like [PHP](https://en.wikipedia.org/wiki/PHP), [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C++), [HTML5](https://en.wikipedia.org/wiki/HTML5),[[3]](https://en.wikipedia.org/wiki/NetBeans#cite_note-3) and [Javascript](https://en.wikipedia.org/wiki/JavaScript). Applications based on NetBeans, including the NetBeans IDE, can be extended by [third party developers](https://en.wikipedia.org/wiki/Third_party_developer).

**5.2.2 Java**

Java technology is both a programming language and a platform. The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed.

**5.2.3 JAVA SERVER PAGES (JSP)**

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and a mature re-usable component model. The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headaches; it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other’s work.

**5.2.3.1 Features of JSP**

**Portability:**

Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition, translation, and management of the Java Server Page lifecycle and its interaction components.

**Components:**

It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components current supported include Java Beans, and Servlets.

**Processing:**

A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.

**Access Models:**

A Java Server Pages file may be accessed in at least two different ways. A client’s request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client.

In both of the above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.

**5.2.3.2 Steps in the execution of a JSP Application:**

The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.

This request is transferred to the JavaWebServer. At the server side JavaWebServer receives the request and if it is a request for a jsp file server gives this request to the JSP engine.

JSP engine is program which can understand the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the JavaWebServer and then it is transferred back to the result is given back to the JavaWebServer and then it is transferred back to the client.

**5.2.4 SERVLETS**

**5.2.4.1 Introduction**

The Java web server is Java Soft’s own web Server. The Java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers for any Internet or Intranet client/server system. Servlets are to a web server, how applets are to the browser.

**5.2.4.2 Servlets**

Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side-object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, pluggable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example, an HTTP Servlets can be used to generate dynamic HTML content.

When you use Servlets to do dynamic content you get the following advantages:

* They’re faster and cleaner than CGI scripts
* They use a standard API (the Servlets API)
* They provide all the advantages of Java (run on a variety of servers without needing to be rewritten).

**5.2.4.3 Attractiveness of Servlets**

There are many features of Servlets that make them easy and attractive to use.

These include:

Easily configured using the GUI-based Admin tool

Can be loaded and invoked from a local disk or remotely across the network.

Can be linked together, or chained, so that one Servlets can call another Servlets, or several Servlets in sequence.

Can be called dynamically from within HTML pages, using server-side include tags.

Are secure - even when downloading across the network, the Servlets security model and Servlets sandbox protect your system from unfriendly behavior.

**5.2.5 HYPER TEXT MARKUP LANGUAGE**

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive.

HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

**Basic HTML Tags:**

< ! -- --> specifies comments

<A>.......... </A> Creates hypertext links

<B>.......... </B> Formats text as bold

<BIG>.......... </BIG> Formats text in large font.

<BODY>... </BODY> Contains all tags and text in the HTML document

<CENTER>... </CENTER> Creates text

<DD>... </DD> Definition of a term

<DL>...</DL> Creates definition list

<FONT>...</FONT> Formats text with a particular font

<FORM>...</FORM> Encloses a fill-out form

<FRAME>...</FRAME> Defines a particular frame in a set of frames

<H#>...</H#> Creates headings of different levels

<HEAD>...</HEAD> Contains tags that specify information about a document

<HR>...</HR> Creates a horizontal rule

<HTML>...</HTML> Contains all other HTML tags

<META>...</META> Provides meta-information about a document

<SCRIPT>...</SCRIPT> Contains client-side or server-side script

<TABLE>...</TABLE> Creates a table

<TD>...</TD> Indicates table data in a table

<TR>...</TR> Designates a table row

<TH>...</TH> Creates a heading in a table

**5.2.5.1 ADVANTAGES**

A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.

**5.2.6 MYSQL**

**MYSQL 8.0:**

MySQL is an [open-source](https://en.wikipedia.org/wiki/Open-source) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS). The MySQL development project has made its [source code](https://en.wikipedia.org/wiki/Source_code) available under the terms of the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), as well as under a variety of [proprietary](https://en.wikipedia.org/wiki/Proprietary_software) agreements. MySQL was owned and sponsored by a single [for-profit](https://en.wikipedia.org/wiki/Business) firm, the [Swedish](https://en.wikipedia.org/wiki/Sweden) company [MySQL AB](https://en.wikipedia.org/wiki/MySQL_AB), now owned by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation).

**5.2.6.1 Introduction**

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. This tutorial will give you a quick start to MySQL and make you comfortable with MySQL programming.

## 5.2.6.2 MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

* MySQL is released under an open-source license. So you have nothing to pay to use it.
* MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
* MySQL uses a standard form of the well-known SQL data language.
* MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
* MySQL works very quickly and works well even with large data sets.
* MySQL is very friendly to PHP, the most appreciated language for web development.
* MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
* MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

**5.3 Sample Code**

<?xml version="1.0"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>MEDICAL SHOP</title>

<style type="text/css" media="all">

@import "global.css";

</style>

</head>

<body>

<div id="wrapper">

<div id="header">

<div id="logobox">

<img src="img/Pics.jpg" width="220" height="150"></img>

<h2></h2>

</div>

<div><img src="img/11.jpg" width="720" height="160"></img></div>

</div>

<div id="topnavi">

<div class="spacing1">

<ul>

<li><a href="shophome.jsp" title="item">HOME</a></li>

<li class="selected"><a href="shop\_addmedicine.jsp" title="item">Add Medicine</a></li>

<li><a href="shop\_updatemedicine.jsp" title="item">Update Medicine</a></li>

<li><a href="shop\_deletemedicine.jsp" title="item">Delete Medicine</a></li>

</ul>

</div>

</div>

<div id="bodybox">

<div id="subnavi">

<ul>

<li><a href="shophome.jsp" title="item">Home</a></li>

<li class="selected"><a href="shop\_addmedicine.jsp" title="item">Add Medicine</a></li>

<li><a href="shop\_userpurchased.jsp" title="item">User Purchased Medicines</a></li>

<li><a href="index.html" title="item">Logout</a></li>

</ul>

</div>

<div id="content">

<center>

<h1>Add Medicine Details</h1>

<br><br>

<form name="myform" id="loginForm" action="shop\_addmedicine1.jsp" method="post" onsubmit="return validateform()" >

<center>

<table>

<tr>

<th><font color="black" size="4">Medicine Type :</th>

<th><input type="text" name="mtype" placeholder= "Medicine Type" style="height:30px; width:170px"></input>

</th>

</tr>

<tr>

<th><font color="black" size="4">Medicine Name :</th>

<th><input type="text" name="mname" placeholder= "Medicine Name" style="height:30px; width:170px"></input>

</th>

</tr>

<tr>

<th><font color="black" size="4">Company Name </th>

<th><input type="text" name="cname" placeholder= "Company Name" style="height:30px; width:170px"></input>

</th>

</tr>

<tr>

<th><font color="black" size="4">Quantity: </th>

<th><input type="text" name="quantity" placeholder= "Quantity" style="height:30px; width:170px"></input>

</th>

</tr>

<tr>

<th><font color="black" size="4">Price: </th>

<th><input type="text" name="price" placeholder= "Price" style="height:30px; width:170px"></input>

</th>

</tr>

<tr>

<th><font color="black" size="4">Date of Manufacture:</th>

<th><input type="date" name="dom" placeholder= "Date of Manufacture" style="height:30px; width:170px"></input>

</th>

</tr>

<tr>

<th><font color="black" size="4">Date of Exp:</th>

<th><input type="date" name="doe" placeholder= "Date of Exp" style="height:30px; width:170px"></input>

</th>

</tr>

<tr>

<td>

<input type="submit" value="ADD" style="height:30px; width:65px"/>

</td>

</tr>

</table>

</center>

</form>

<br><br><br><br><br><br><br>

</body>

<footer><h2>Copyright © 2018 Appboy Studios.All rights reserved.</h2></footer>

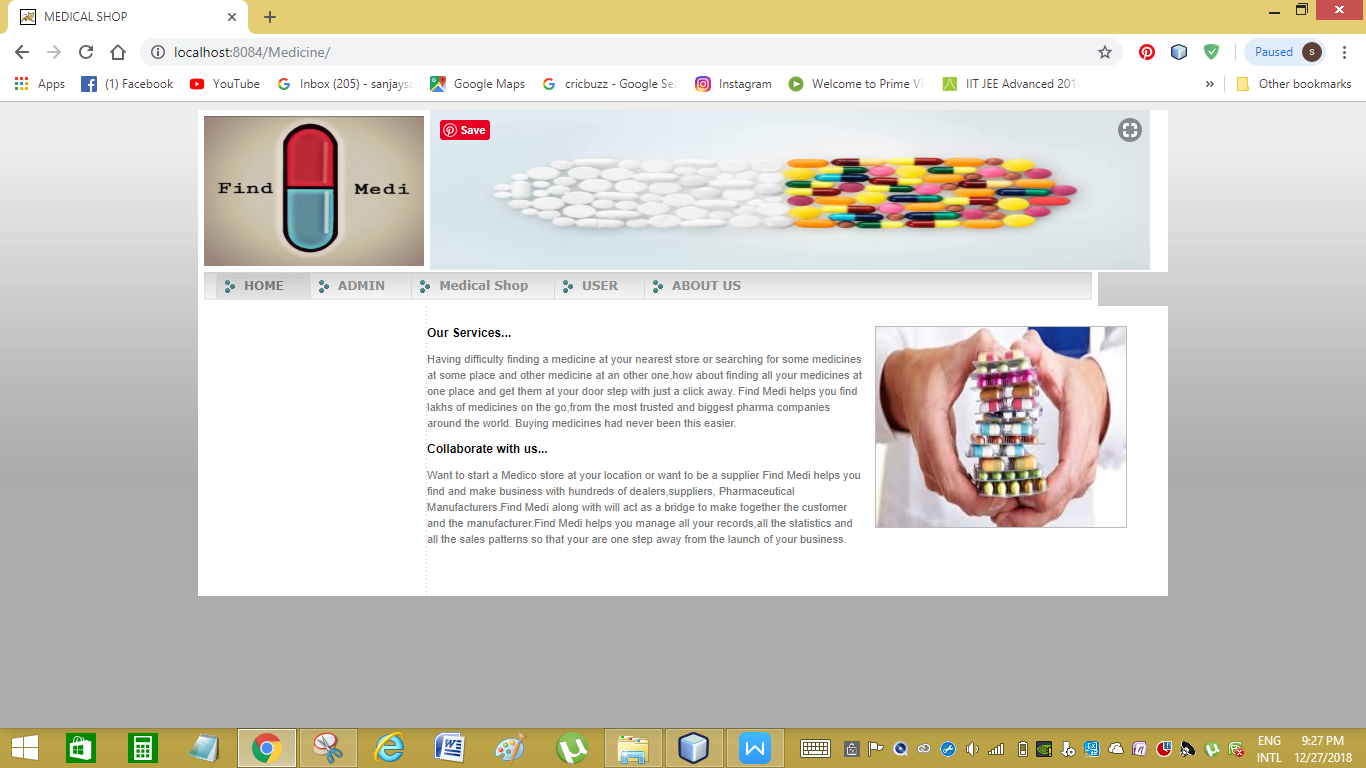
</html>

**File name:**shop\_addmedicine.jsp

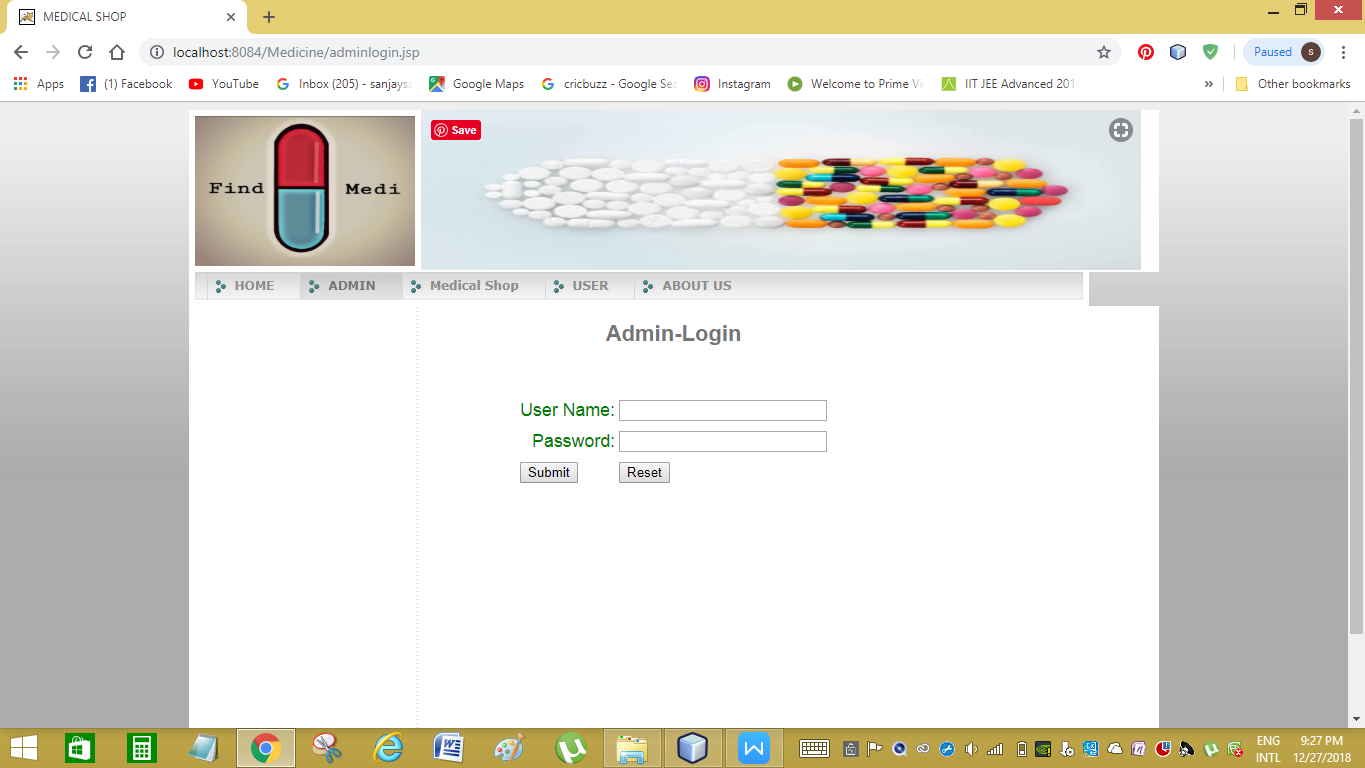
**Operation of the file :** In the above JSP file we see that the dealer adds new medicines to make them available for the customer which will be saved in the server. These Details will be used at the time of purchase.

**5.4 Results**

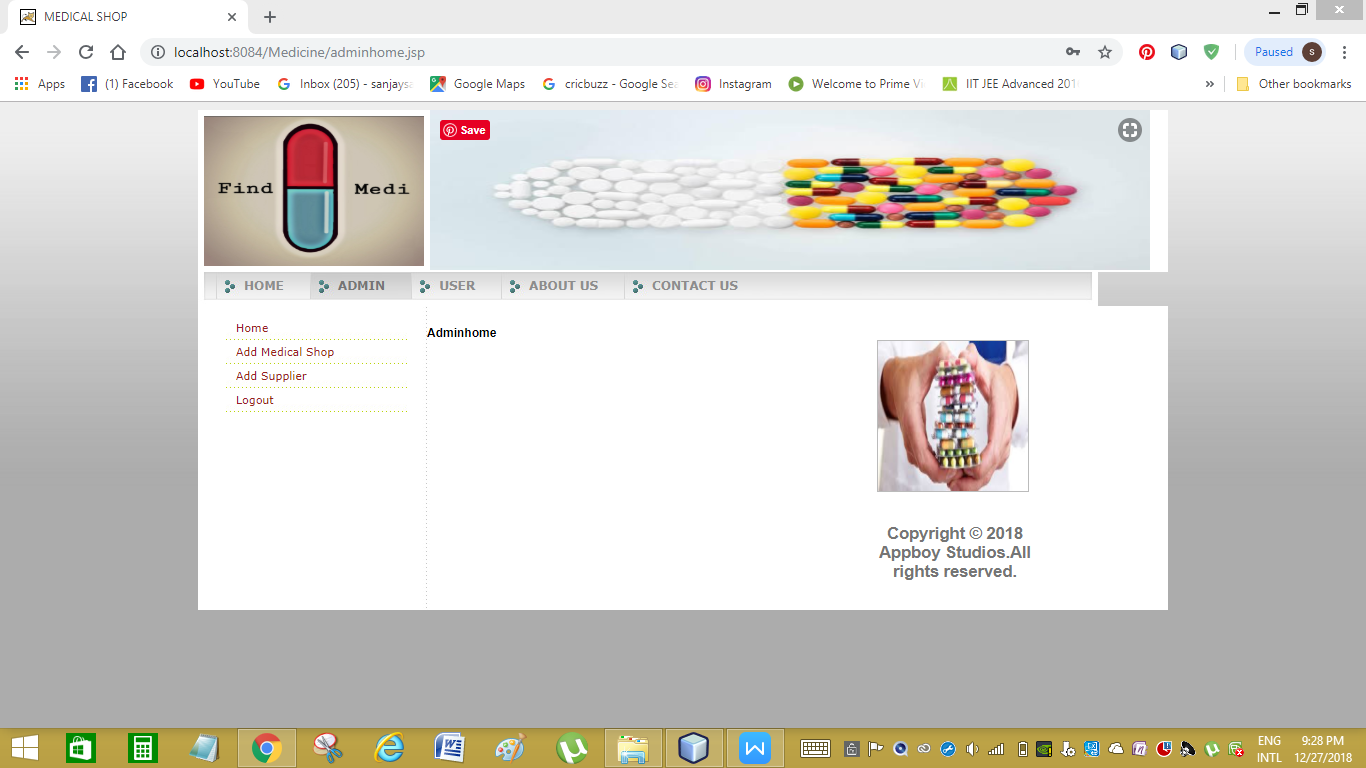
**5.4.1 Output Screens**

****

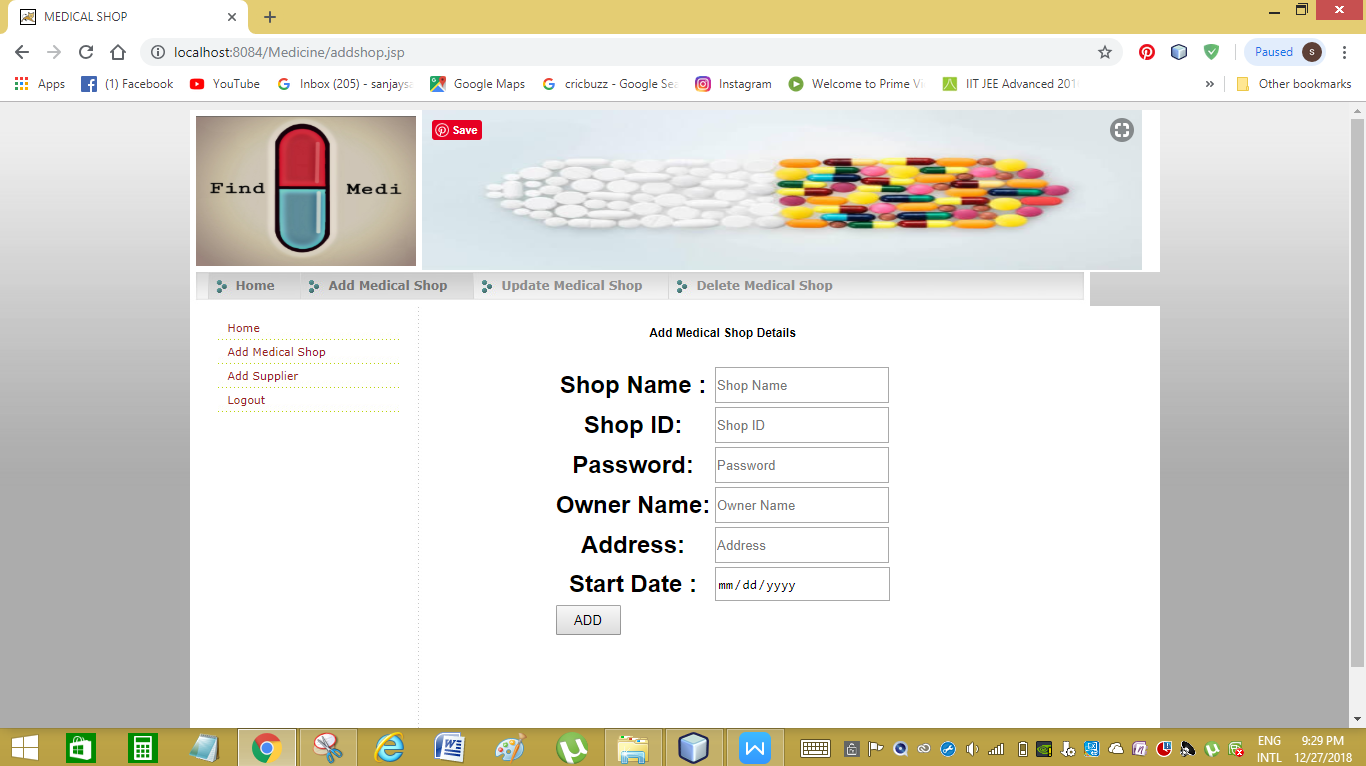
**Fig 5.1 Home Page**

****

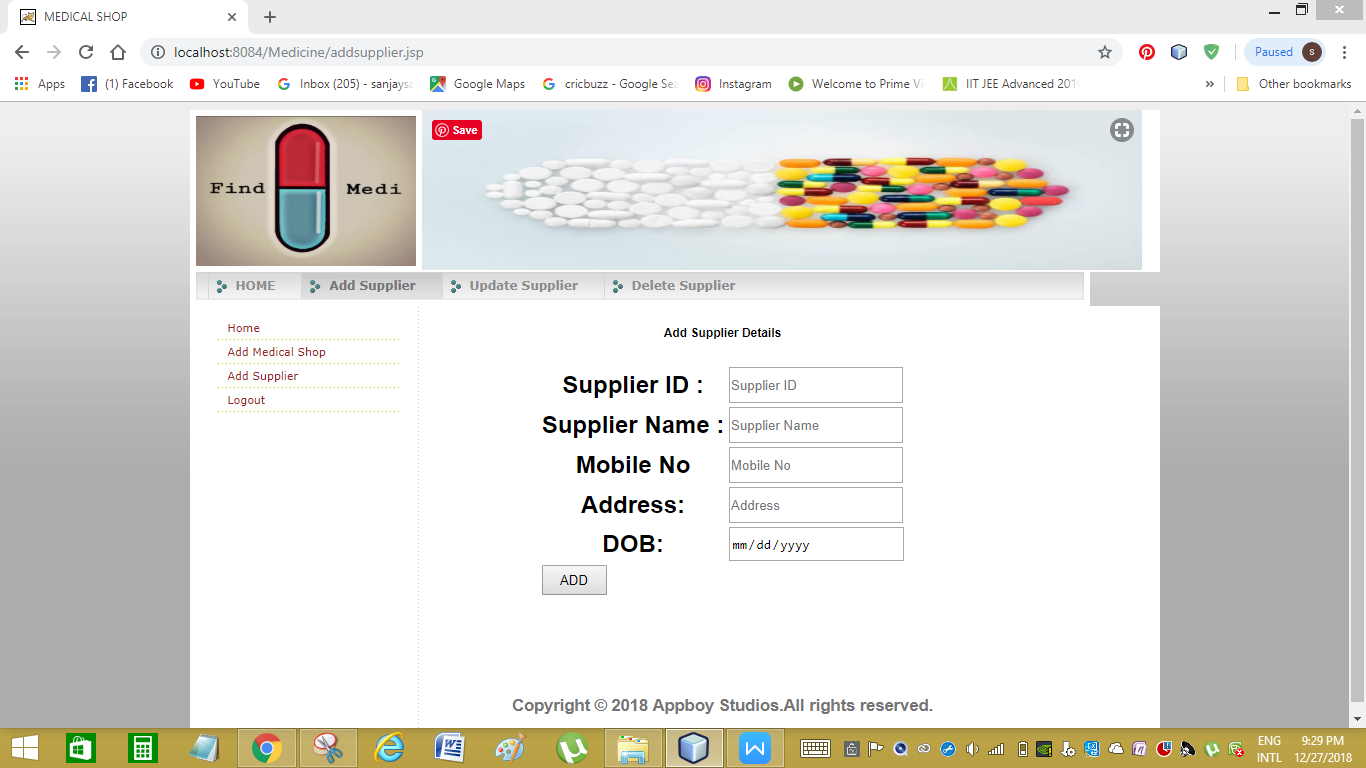
**Fig 5.2 Admin Login**

****

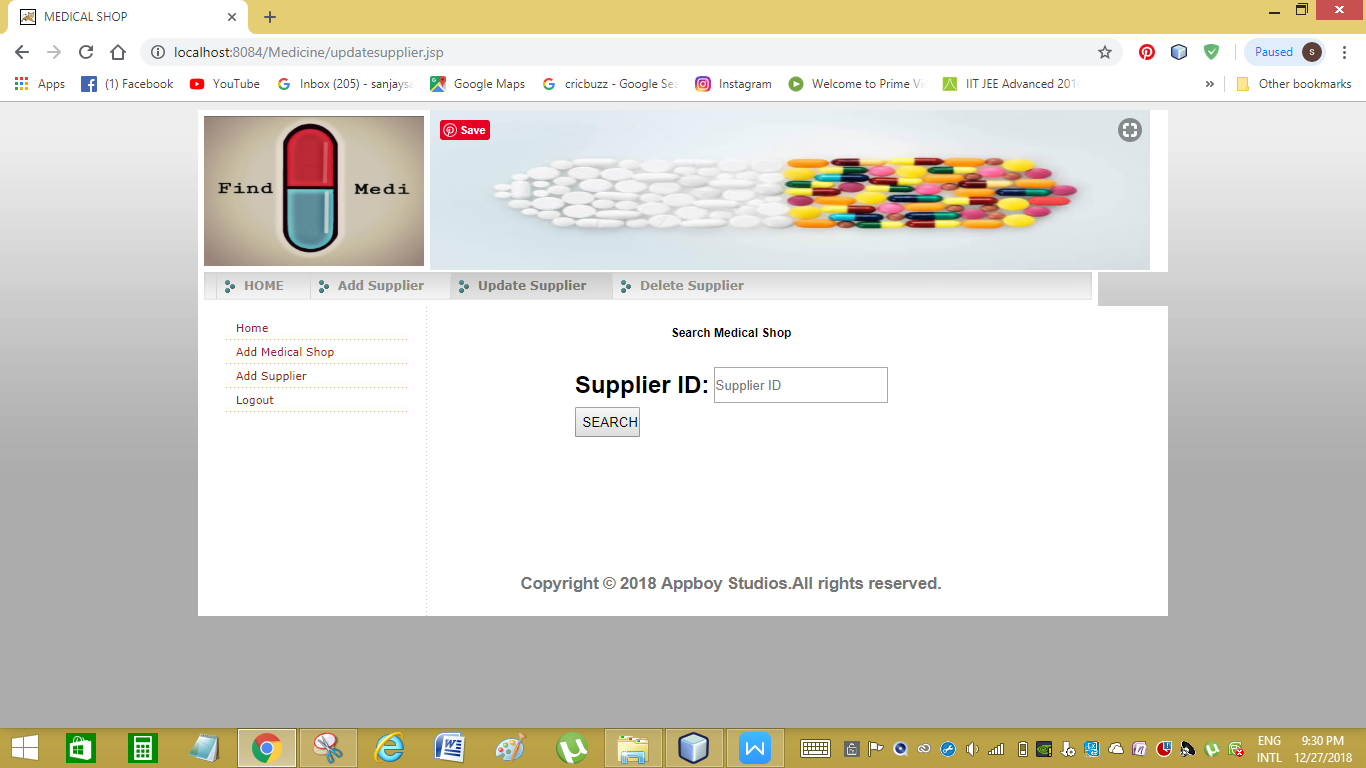
**Fig 5.3 Admin Home**

****

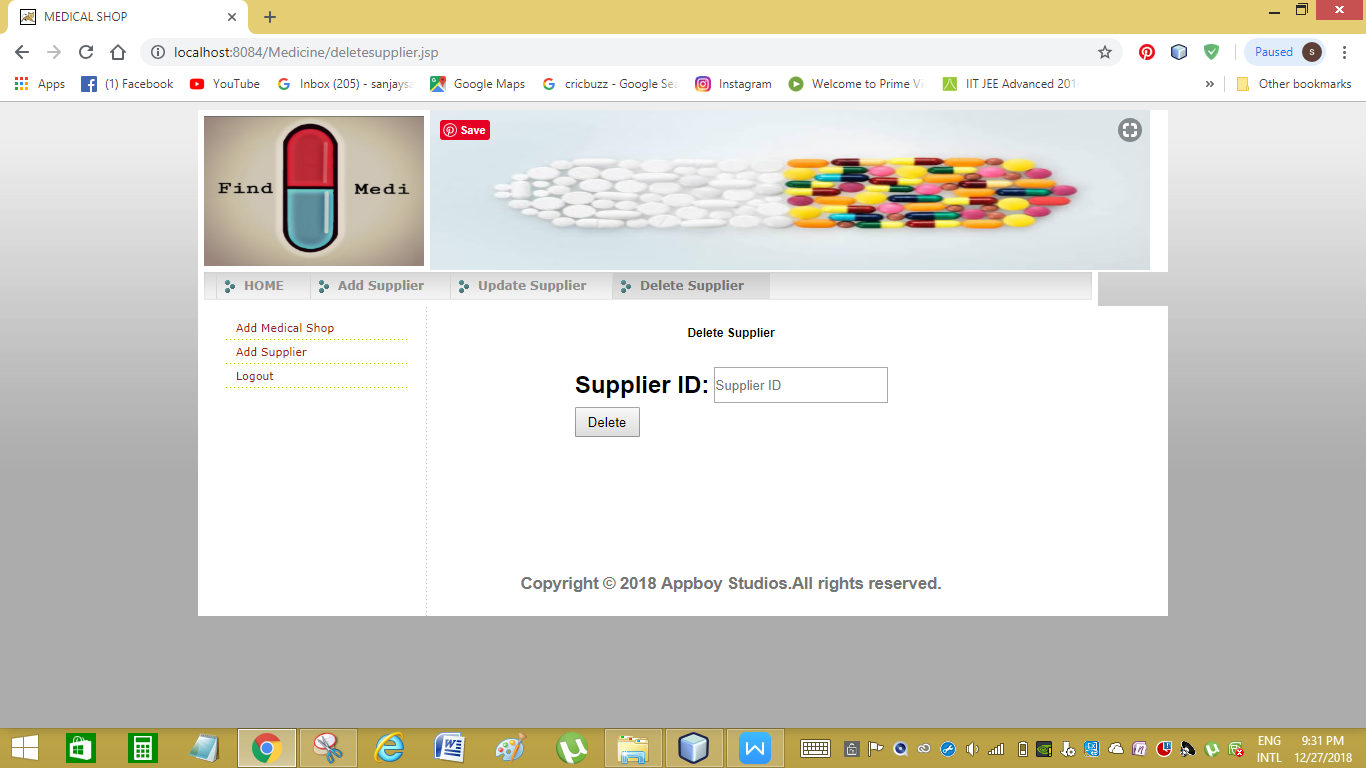
**Fig 5.4 Admin Add Medical Shop**

****

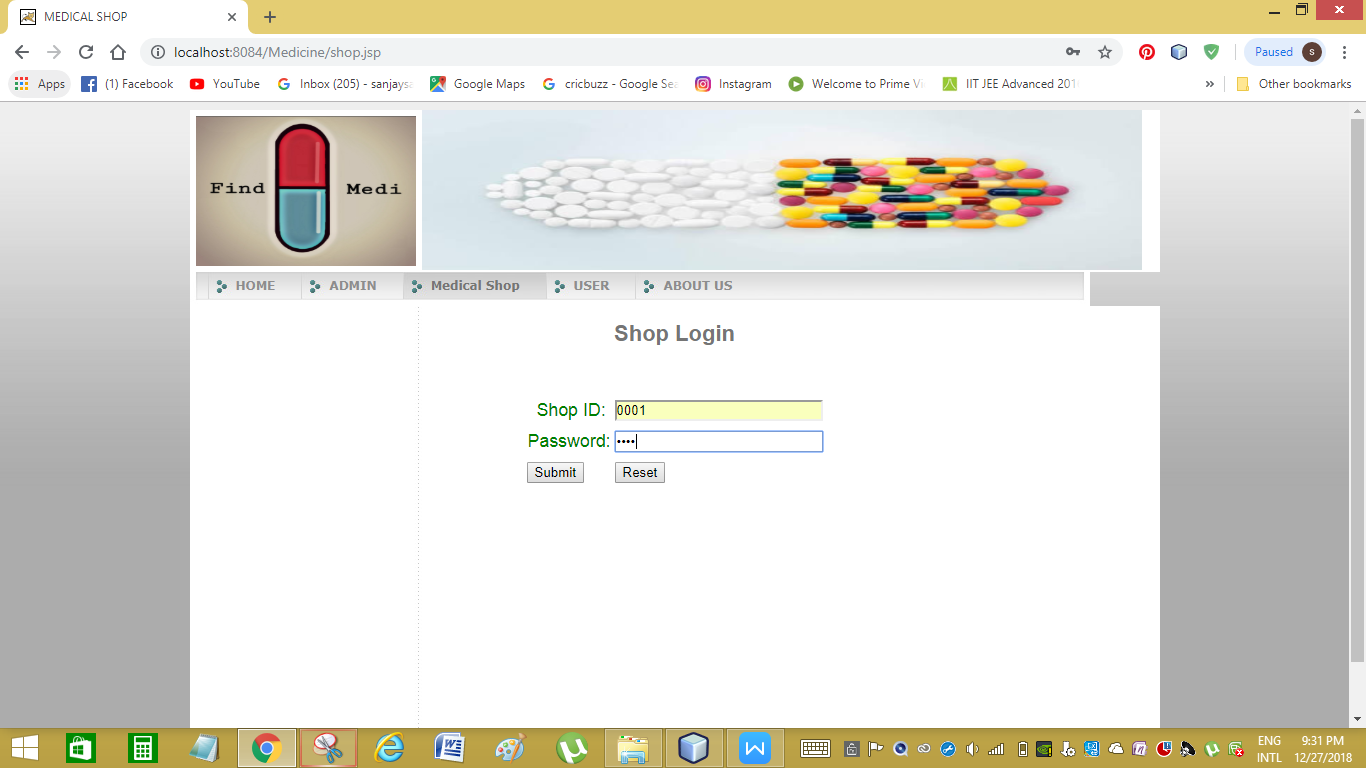
**Fig 5.5 Admin Add Supplier**

****

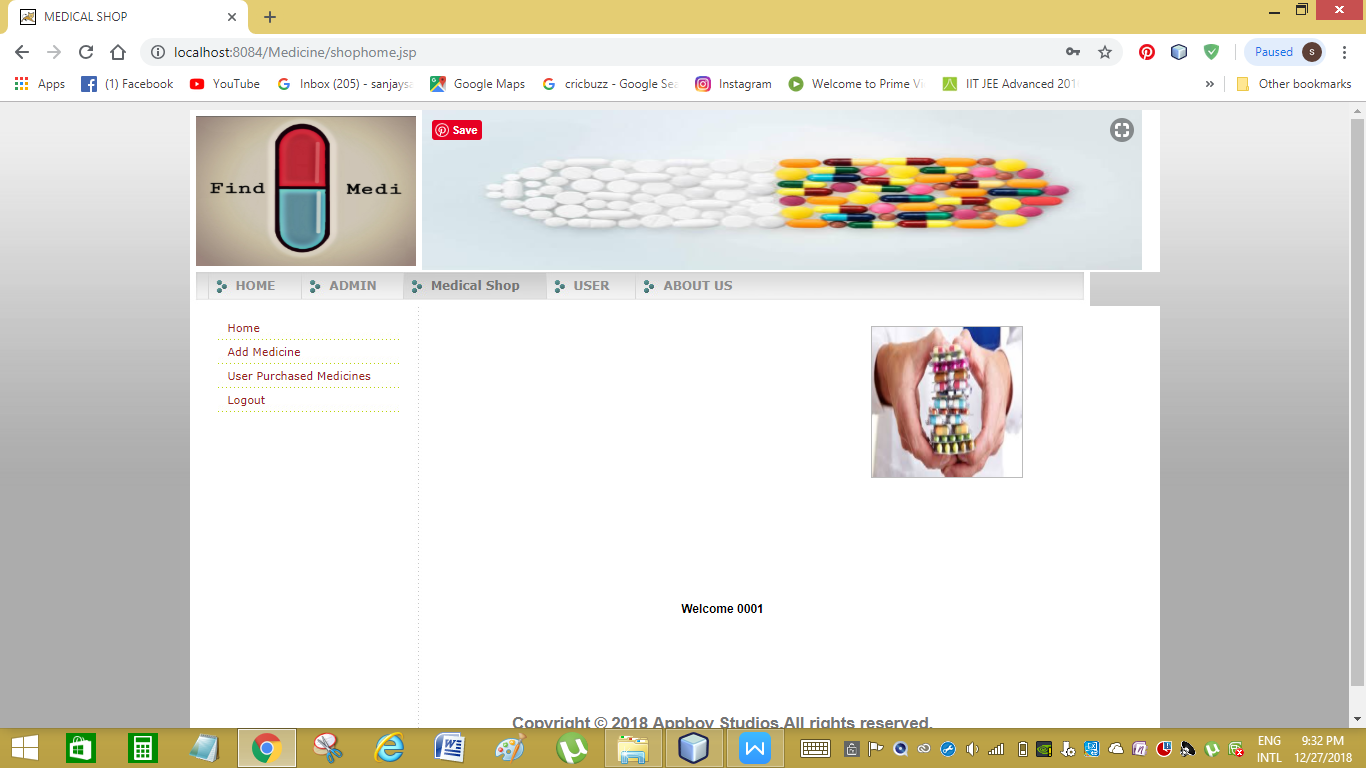
**Fig 5.6 Update Medical Shop**

****

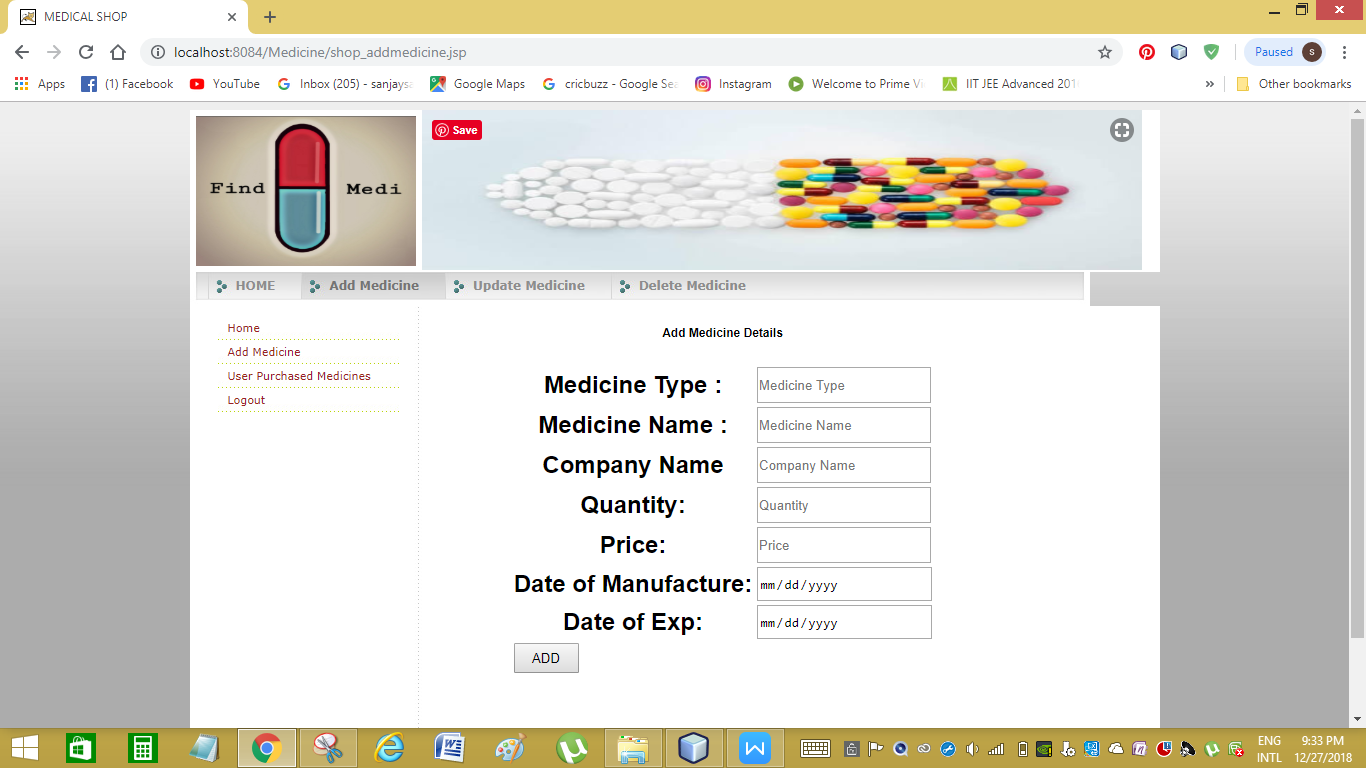
**Fig 5.7 Delete Medical Shop**

****

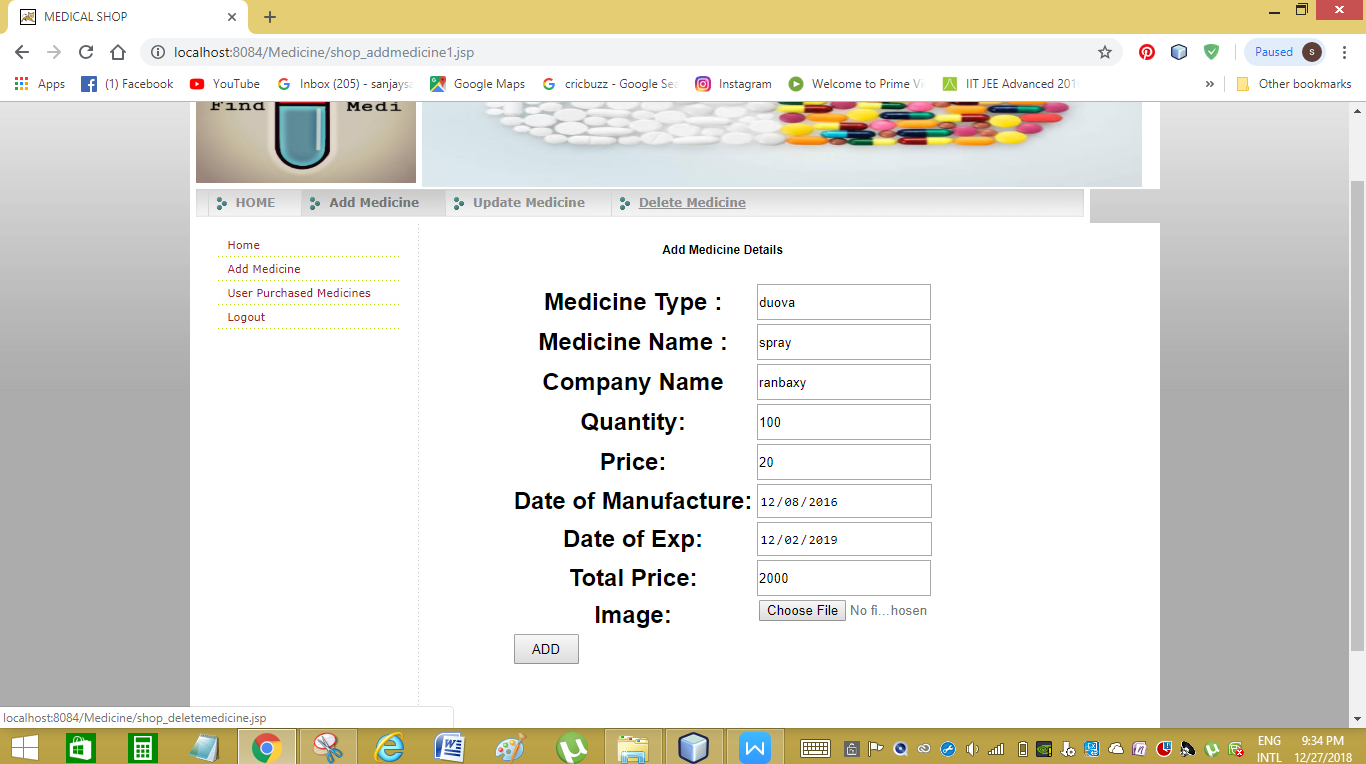
**Fig 5.8 Medical Shop Login**

****

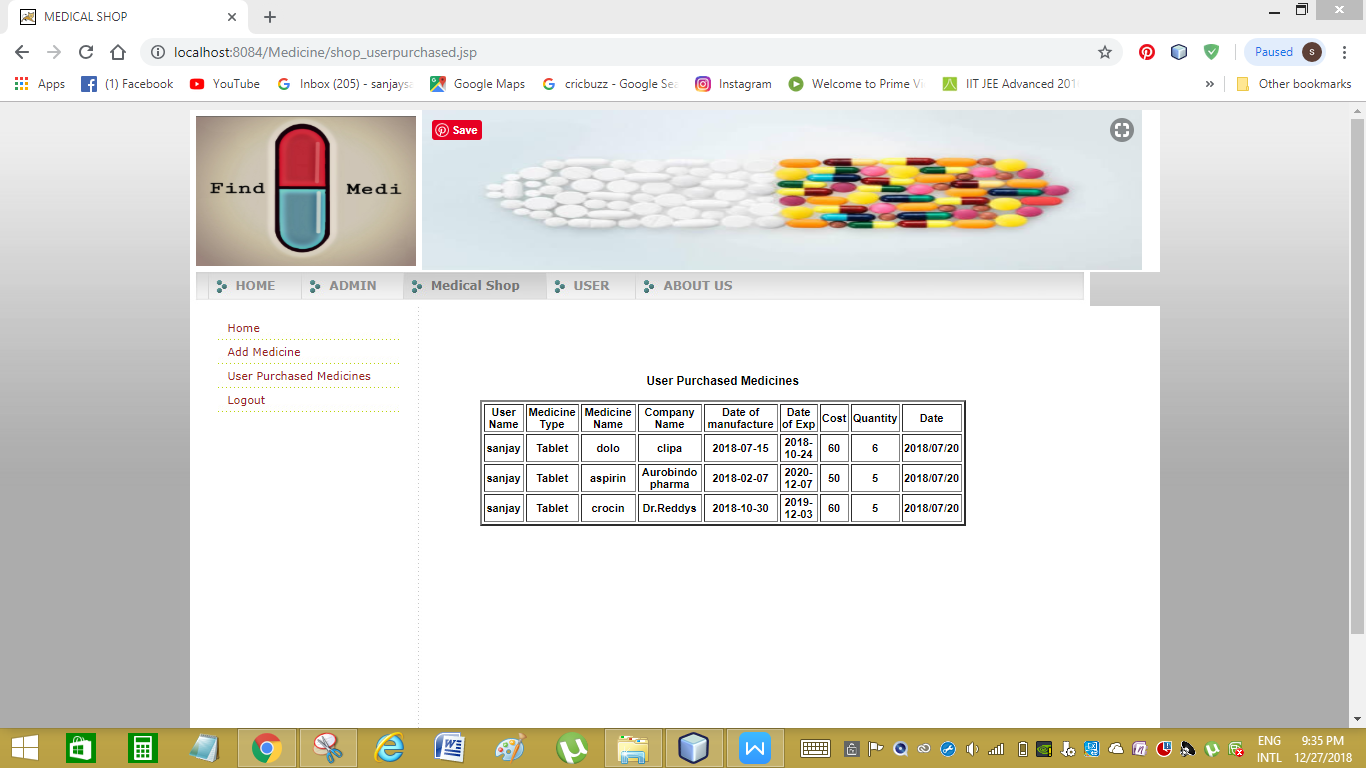
**Fig 5.9 Medical Shop Home**

****

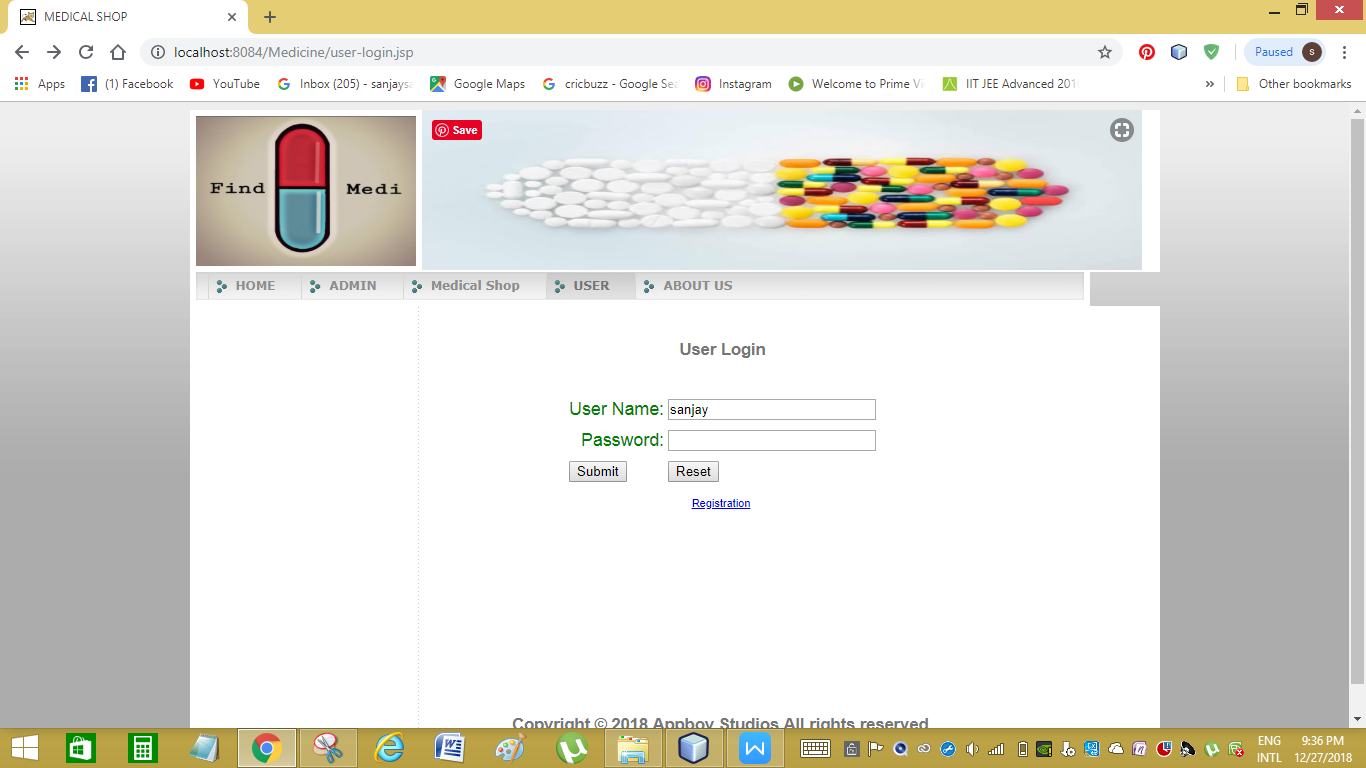
**Fig 5.10 Add Medicine**

****

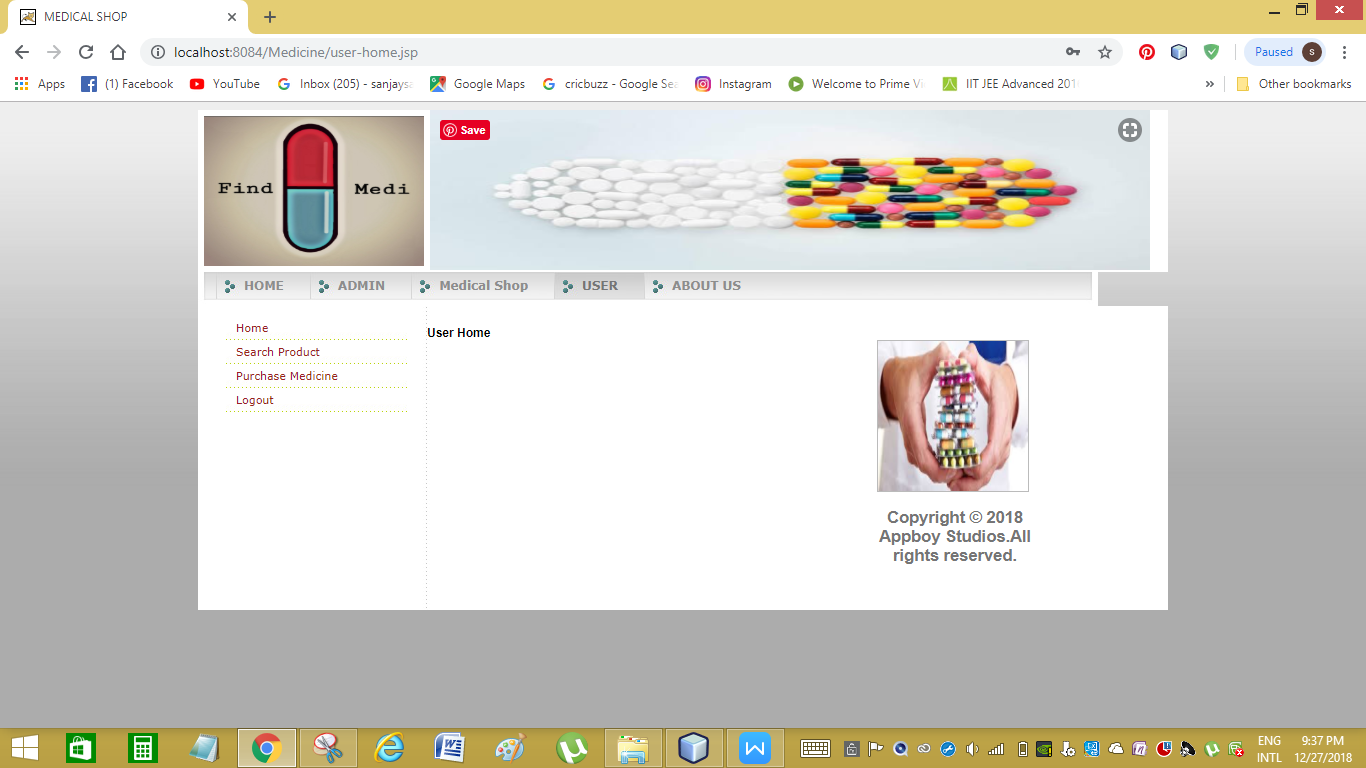
**Fig 5.11 Medicine Details**

****

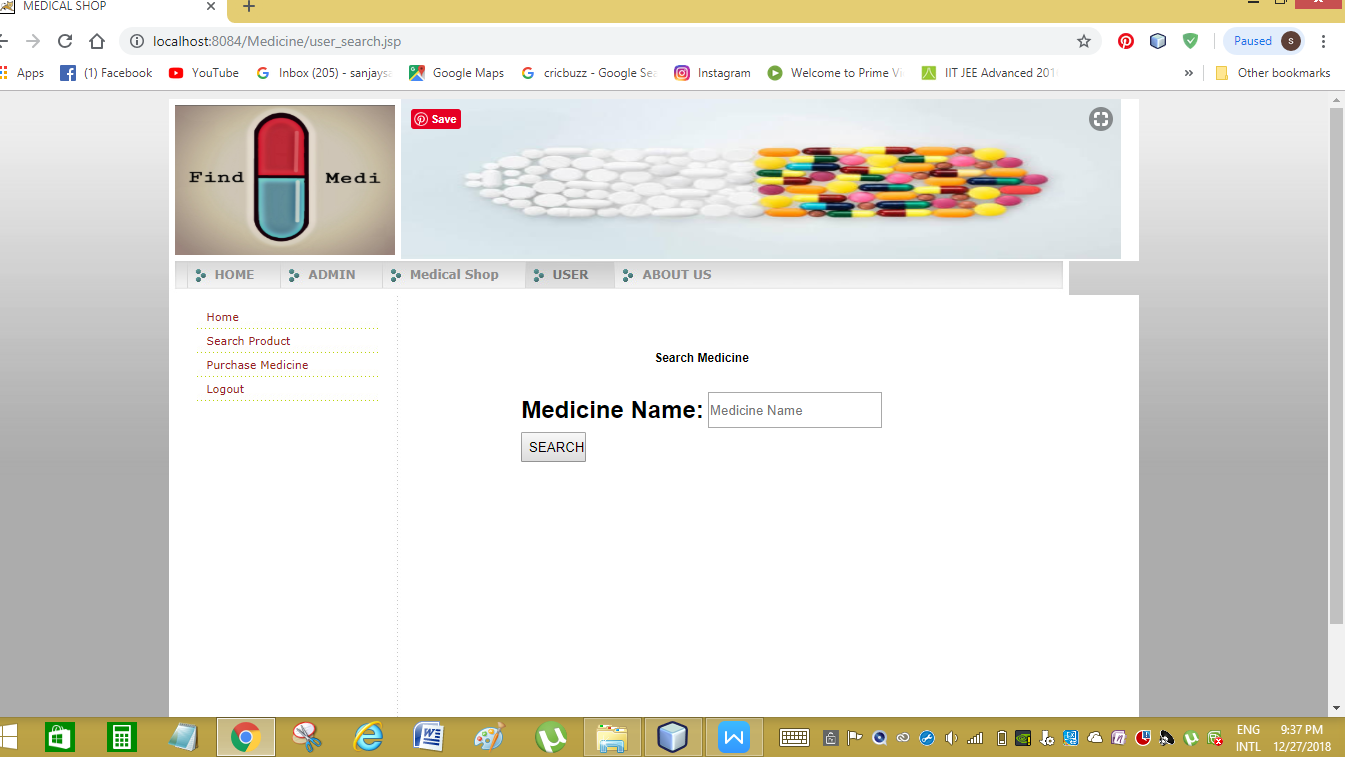
**Fig 5.12 User Purchased Medicines**

****

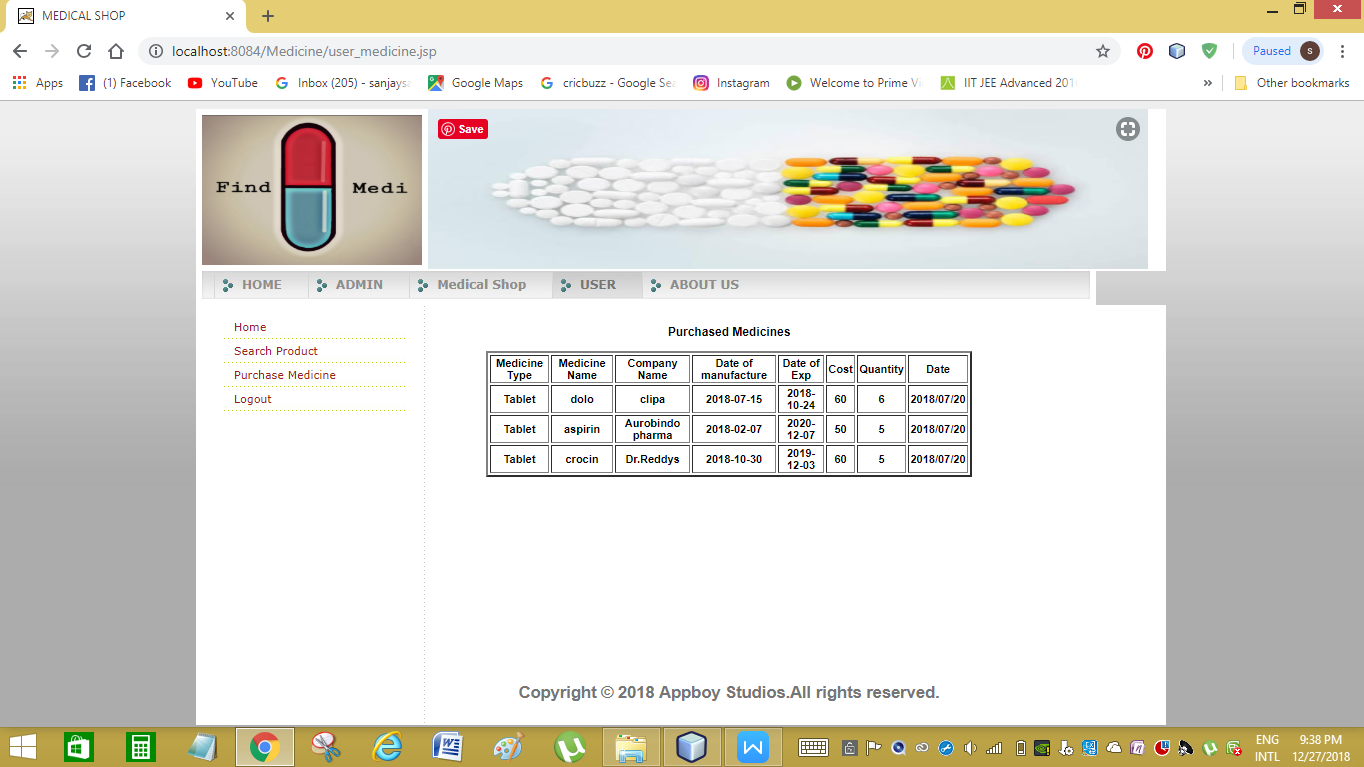
**Fig 5.13 User Login**

****

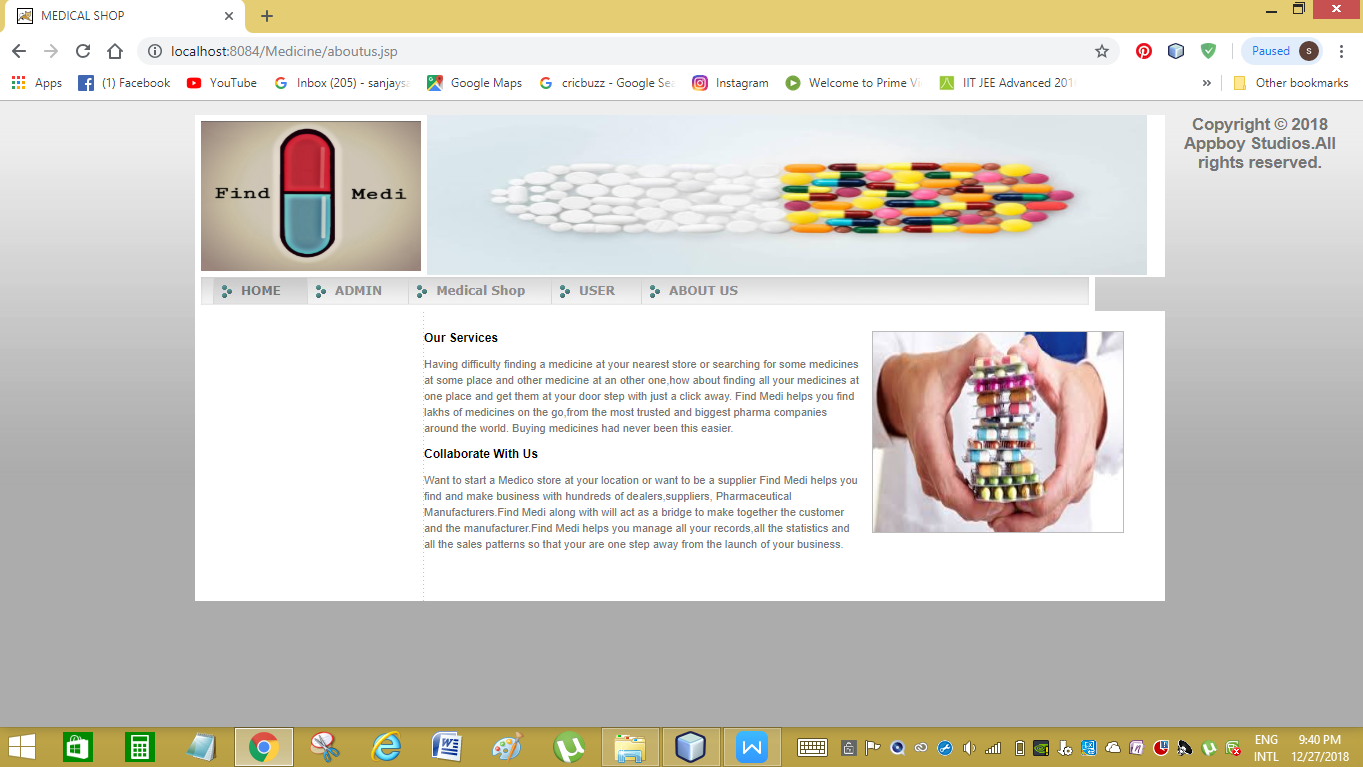
**Fig 5.14 User Home**

****

**Fig 5.15 Search Medicine**

****

**Fig 5.16 User Purchase Medicine**

****

**Fig 5.17 About Page**

**6. TESTING**

**6.1 INTRODUCTION**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of testing. Each test type addresses a specific testing requirement.

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

**6.1.1 TESTING OBJECTIVES**

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

* Testing is a process of executing a program with the intent of finding an error.
* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable standards.

**6.2 Levels of Testing:**

In order to uncover present in different phases we have the concept of levels of testing.

The basic levels of Testing:

Client needs acceptance testing

Requirements system testing

Design integration testing

Code unit testing

**Fig 6.1 Levels of Testing**

**6.2.1** **Code Testing:**

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

**6.2.2 Specification Testing:**

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

**6.2.3** **Unit testing:**

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

**6.2.4 BLACK BOX TESTING**

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.



**Fig 6.2 Black box Testing**

The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

* Initially requirements and specifications of the system are examined.
* Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
* Tester determines expected outputs for all those inputs.
* Software tester constructs test cases with the selected inputs.
* The test cases are executed.
* Software tester compares the actual outputs with the expected outputs.
* Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

* Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
* Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements  such as performance, scalability, usability.
* Regression testing – Regression testing is done  after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

**6.2.5 WHITE BOX TESTING**

White Box Testing is the testing of a software solution's internal coding and infrastructure.It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

## What do you verify in White Box Testing ?

White box testing involves the testing of the software code for the following:

* Internal security holes
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

## How do you perform White Box Testing?

  To give you a simplified explanation of white box testing, we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique:

**STEP 1) UNDERSTAND THE SOURCE CODE**

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

**Step 2) CREATE TEST CASES AND EXECUTE**

The second basic step to white box testing involves testing the application’s source code for proper flow and structure. One way is by writing more code to test the application’s source code. The tester will develop little tests for each process or series of processes in the application. This  method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

**6.2.6 System testing:**

Once the individual module testing is completed, modules are assembled and integrated to perform as a system. The top down testing, which began from upper level to lower level module, was carried out to check whether the entire system is performing satisfactorily.

There are three main kinds of System testing:

1. Alpha Testing
2. Beta Testing
3. Acceptance Testing

**Alpha Testing**:

This refers to the system testing that is carried out by the test team with the Organization.

**Beta Testing**:

This refers to the system testing that is performed by a selected group of friendly customers

**Acceptance Testing**:

This refers to the system testing that is performed by the customer to determine whether or not to accept the delivery of the system.

**Table 6.1. Acceptance Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Condition ID | Description of coverage | Expected results | Covered by script |
| 1. | Verification of a particular record | If a particular record already exists it displays a message | This type of test in {verify} procedure in every Jsp file where a record is inserted via an interface |
| 2. | Updating of a particular record | All the details should not be updated. | This type of test is covered in all the Asp files where updations are made. |
| 3. | Validity of login | Only the authorized persons must access system. | This is covered in the login procedure for the validity of a user |

**6.2.7 Integration Testing**:

Data can be lost across an interface, one module can have an adverse effort on the other sub functions, when combined, may not produce the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. The testing was done with sample data. The developed system has run successfully for this sample data. The need for integrated test is to find the overall system performance.

**6.2.8 Output Testing**:

After performance of the validation testing, the next step is output testing. The output displayed or generated by the system under consideration is tested by asking the user about the format required by system. The output format on the screen is found to be correct as format was designed in the system phase according to the user needs. Hence the output testing does not result in any correction in the system.

**6.3 Test Plan**:

The test-plan is basically a list of testcases that need to be run on the system. Some of the testcases can be run independently for some components (report generation from the database, for example, can be tested independently) and some of the testcases require the whole system to be ready for their execution. It is better to test each component as and when it is ready before integrating the components. It is important to note that the testcases cover all the aspects of the system (ie, all the requirements stated in the RS document).

**7. CONCLUSION AND FUTURE ENHANCEMENT**

**7.1 Conclusion**

As the technology grows day to day, there are lots of changes happening throughout entire system and importantly connecting technology with Medication and Health is very much important. By proving a way for buying the medicines online from a preferred shop and maintaining the previous medical records and medical shops managing their records efficiently helps in better customer as well as managing experience.

**7.2 Future Enhancement**

* In the future we are planning to provide a facility for the each shop owner under the same name, where they can work together under one roof.
* In the future we intend to provide a facility for customers in by which he can maintain all the medical records and all their ongoing medical treatments with the help of notifications and messaging and auto remainders help in reducing the stress of remembering what to and when to buy.

**REFERENCES**

# MySQL Explained: Your Step-by-Step Guide

# by [Andrew Comeau](https://www.amazon.in/Andrew-Comeau/e/B00FNZRDGQ/ref=dp_byline_cont_ebooks_1)  (Author), [Stephen Burge](https://www.amazon.in/Stephen-Burge/e/B004GIFQX8/ref=dp_byline_cont_ebooks_2)  (Editor)

# Programming Windows Store Apps with HTML, CSS, and JavaScript (Developer Reference)

by [Kraig Brockschmidt](https://www.amazon.in/Kraig-Brockschmidt/e/B001JRXR7Q/ref=dp_byline_cont_ebooks_1)  (Author)

1. Brand & Generic Drugs Guide Paperback – 21 Sep 2011

by [Jones & Bartlett Learning](https://www.amazon.in/s/ref=dp_byline_sr_book_1?ie=UTF8&field-author=Jones+&+Bartlett+Learning&search-alias=stripbooks) (Author)

# Web Links:

-> <https://www.w3schools.com/>

-> <https://www.javatpoint.com/>

-> <https://www.drugs.com/drug_information.html>