Online Medical Store Management System

A Project Work

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

Computer science and business systems

Submitted by:

CHIRAG (20CBS1049), SAHIL SACHDEVA (20CBS1046)

PARTH SAHNI (20CBS1047), TASHU CHUGH (20CBS1058)

Under the Supervision of:

Mr. Ashish Kumar Rastogi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

APEX INSTITUE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

May 2022

DECLARATION

We, as a team**,** student of 'Bachelor in Engineering in Computer science and business systems' session 2020 – 2024, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled '**Online Medical store management system'** is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Online Medical management system (Group 5)

UIDs: 20CBS1049, 20CBS1046, 20CBS1047, 20CBS1058

Date: 7 May 2022

Place: Chandigarh University

ABSTRACT

The Online Medical Store Management System is a techno-designed solution to the problems that professionals at medical stores generally face. The problems faced by professionals lead to wastage of money, time and resources. Here, the resources can be defined as the essential products used in the manufacturing of medicine. These resources get wasted as lots of medicines get wasted every year due to improperly maintained stocks in the medical store.

So, the solution to all these problems is the online medical store management system that will help professionals in keeping their record of their medicine stock efficiently and properly. These records are maintained digitally, so any medicine or stock is easy to find in the records. Secondly, these records get maintained by the exact details of the issue date, expiry date, precautions, side effects, etc. so we can conclude that this will be something different in this field as this will help in maintaining the customer's trust.

Thus, our system includes:

1. It helps in unique identification of the medicine with the help of the reference number.
2. Saving the data of the medicine related to its issue date, expiry date, precautions, dosage, quantity available and price in the database.
3. To uniquely identify each product and its dealer with the help of unique lot number allotted to each bill.
4. It also provides you a about us page to the users that is a quick guide to the system.

So, we are on aim to make the best user-friendly system for better utilization and performance. For more interaction, we here provide a heath tip option and login system that will definitely provide authentication to your system.

ACKNOWLEDGEMENT

Place: CHANDIGARH UNIVERSITY

Date: 7 May 2022

I would like to express my profound gratitude to Mr. Vinay Sharma, Program leader of AIT-CSE (Apex institute of technology) department for their contributions to the completion of my project titled **Online Medical Store Management System**.

I would like to express my special thanks to our mentor Mr. **Ashish Kumar Rastogi** for his time and efforts he provided throughout the semester. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to you.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

CHIRAG (20CBS1049)
SAHIL SACHDEVA (20CBS1046)
PARTH SAHNI (20CBS1047)
TASHU CHUGH (20CBS1058)

TABLE OF CONTENTS

	Title page	1
	Declaration of the student	3
	Abstract	4
	Acknowledgement	5
	Table of contents	6
	List of figures	7
	List of tables	8
	Timeline / Gantt chart	9
1.	**Introduction**	11-15
	1.1 Problem Formulation	
	1.2 Project overview/ Specification	
	1.3 Hardware Specification	
	1.4 Software Specification	
	1.3.1	
	1.3.2	
	…	
2.	**Literature Survey**	16-20
	2.1 Existing system	
	2.2 Proposed system	
	2.3 Feasibility study	
3.	**Problem formulation**	21
4.	**Objectives**	22
5.	**Methodology**	23-39
6.	**Conclusions and discussions**	40-100
7.	**References**	101

LIST TO FIGURES

Sr. No.	Figures	Page no.
1	DFDs	23
2	Use case diagrams	26
3	Class diagram	28
4	Component diagram	30
5	Sequence diagram	32
6	Activity diagram	34
7	Deployment diagram	36
8	ER Diagrams	38
9	Landing page	60
10	Login pages	60-62
11	Medicine Add department pages	64-71
12	Medicine function pages	73-82
13	Medicine database pages	83-90
14	Health tips pages	91-95
15.	About us pages	97-99

LIST OF TABLES

(There is no table present in this report)

INTRODUCTION

1.1. PROBLEM DEFINITION:

In the present time, the medical stores are looking for the services that are accurate and reliable for providing services to the customers and workers.

Every store is making efforts to computerize their activities for providing better services to the customers.

The medical store management system is the system used for medicine stock inventory in the medical stores.

This system enables the manager of the store to record and manage all activities of the medical shop.

Additional updates:

1. Store Login
2. Notification when a medication is almost out of stock
3. Alert when few medications are supposed to expire.
4. One-click availability for medication delivery. (Further division)

1.2 PROJECT OVERVIEW/ SPECIFICATION:

- ✓ The medical store management system helps in gathering the perfect management in the details.
- ✓ The collection and storage of data with the help of this project will be sensible, obvious and simple.
- ✓ It also helps in understanding the management of the previous time vividly. It also helps in performing the various functions of medical store.
- ✓ The medical store management system is developed to provide ease in documentation, keeping records, maintaining inventory and payment in the medical store which requires proper record keeping and management.
- ✓ The project also helps the manager or administrator of the medical store to assess the stored information in the time of need such as for contacting the suppliers.
- ✓ The project is intended to be the user friendly and interactive in use.
- ✓ The project has a designed in such a way to motivate others to use such as by:
  - ➔ Providing daily health tips
  - ➔ Database show and search options
- ✓ Provided login option for authentication level of purposes.

HARWARE AND SOFTWARE SPECIFICATIONS

Technology Used:

Software or Tools

The tools that I have used in developing this system is:

- Visual Studio Code
- Python Software
- PostgreSQL
- Tkinter
- Microsoft Word for project documentation
- Frameworks
- Windows 98/2000/XP operating system

Hardware Requirements

- A physical memory (RAM) of 512MB and above are required**.**
- Hard disk capacity: 5 GB
- Intel, Celeron or AMD Pentium 3 processor.

This Online Pharmacy Management System project requires and runs on a number of languages and tools which make it fully functionable and better. These include:

- **Visual Studio Code**: It is also referred to as VS Code, it is source code editor made by Microsoft for various platforms like windows, Linux, macOS. It can be used for a mixture of languages like Python, C, C++, and many more. It can be extended by downloading various extensions in that that can make code look better and efficient which are available through a central repository .it also has in-built feature of source control. this feature allows to create repositories as well as to make push, pull requests directly from program. Instead of a undertaking system, it permits customers to open one or greater directories, that may then be stored in workspaces for destiny reuse. This permits it to perform as a language-agnostic code editor for any language. It helps many programming languages and a fixed of functions that

differs in line with language. Visual Studio Code consists of a couple of extensions for FTP, permitting the software program for use as a loose opportunity for internet development. Code may be synced among the editor and the server, with out downloading any more software program.

- **Python:** It is extensively used general-purpose, excessive stage programming language. It turned into created via way of means of Guido van Rossum in 1991 and similarly evolved via way of means of the Python Software Foundation. Python has a easy and established code base, making the updation and upkeep responsibilities of the software program less difficult for builders. They don't require builders to put in writing extra code, and this, in turn, saves them attempt and precious time. Hence, they are able to make use of their time on something greater effective for the organization. It turned into designed with an emphasis on code readability, and its syntax lets in programmers to specific their standards in fewer strains of code. Graphical user interface (GUI) is also what is Python used for. For Using python in visual studio code, we have to download extension in vs code named as Python for basic functionality as well as code runner. These extensions after proper installation allows us to work in python language in visual studio code.

  This project of online pharmacy management system is done in Python language in Tkinter interface section which itself is part of python language. There are in-built directories in python language that helps us in doing the code.

- **Tkinter**: Tkinter is the built-in python module this is used to create GUI programs. It is one of the maximum usually used modules for developing GUI programs in Python because it is straightforward and clean to paintings with. You don't want to fear approximately the setup of the Tkinter module one at a time because it comes with Python already. It offers an object-orientated interface to the Tk GUI toolkit. Tkinter is light-weight and comparatively painless to apply in comparison to different frameworks. This makes it a compelling desire for constructing GUI packages in Python, in particular for packages wherein a current shine is unnecessary, and the pinnacle precedence is to construct something that's

purposeful and cross-platform quickly. In tkinter there is a widget in it. In Tkinter, Widgets are objects; instances of classes that represent buttons, frames, and much more to user to interact more with the application. Tkinter also consists of various standard attributes like dimensions, colors, fonts, anchors and many more. This project is mainly done in tkinter interface that's helps in making our GUI window look great and efficient. For doing this we firstly have to import the Tkinter module and can add widgets in it and after running so that there is a GUI application window made and then can use various widgets and attributes to add more functions in it.

- **GUI**: Graphical User Interface (GUI). It contains of many GUI methods like Tkinter, PyQt, wxPython and many more. But in between all of them Tkinter is one of best toolkit in python that provides best GUI programming.

- **PostgreSQL:** It also known as Postgres, is a free and open-source relational database management highlighting extensibility and SQL compliance. It features transactions with consistency, isolation etc. It is designed to handle a range of workloads, from single machine to data warehouses with many users. It is default database for macOS server and is also available for windows, Linux etc. PostgreSQL consists of integrated binary replication primarily based totally on delivery the changes (write-in advance logs (WAL)) to reproduction nodes asynchronously, with the capacity to run examine-most effective queries in opposition to those replicated nodes. This lets in splitting examine visitors amongst a couple of nodes efficiently. PostgreSQL consists of integrated assist for everyday B-tree and hash desk indexes, and 4 index get entry to methods. PostgreSQL consists of integrated assist for everyday B-tree and hash desk indexes, and 4 index get entry to methods. In it a wide variety of native data types are supported and inheritance and other more features. This tool is designed to work with multiple systems. This tool is used to manage information in the database of both small and large systems.

LITERATURE REVIEW

Literature Review Summary

By reviewing Medical Store Management system, the primary goal which was lacking in the control gadget is that, whilst the shares of the medication receive low or reduced there ought to be an alert displaying the info of the medication ought to be ordered or not.

This small alert is very beneficial due to the fact if the admin forgets to replace the shares and the medication is lacking then there can be lack of the scientific save.

Another important thing is that online medical store offers medicine within the minimum time of 3 hours which is quite large amount of time, what if the patient needs the medicine urgently.
So, in this MEDICAL STORE MANAGEMENT SYSTEM we solved the problem of late transportation.

EXISTING SYSTEM:

The existing system is very intricated to hold track of all records and manage them manually.

As well as, the work is time-consuming and also expensive in this approach report. Work may be not authentic and not the speediest.

The system needs to get computerized to avoid all these limitations and make the tasks getting completed more accurately.

Drawbacks of the existing system

- Time-consuming
- More expensive
- Searching problem
- Maintains problem of all registers
- Less accuracy
- Display multiple reports

PROPOSED SYSTEM

- ✓ The medical store management system helps in gathering the perfect management in the details.
- ✓ The collection and storage of data with the help of this project will be sensible, obvious and simple.
- ✓ It also helps in understanding the management of the previous time vividly. It also helps in performing the various functions of medical store.
- ✓ The medical store management system is developed to provide ease in documentation, keeping records, maintaining inventory and payment in the medical store which requires proper record keeping and management.

FEASIBLITY STUDY

An important objective of conducting the System Analysis is the determination of the feasibility.
Feasibility study is carried out to select the best system that meets the performance requirements.
Feasibility study is necessary to evaluate the feasibility of a project at the earliest possible time.
It is used to investigate the project and examines whether the designed system will be useful to the organization.
Feasibility and risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced.

The different type of feasibility are :

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

Technical Feasibility: Technical Feasibility is used to find out whether the necessary technology, the proposed equipment has the capacity to hold the data, which is used in the project. It is a study of function, performance and constraints that may affect the ability to achieve an acceptance system.

The Technical Feasibility issues usually raised during the feasibility stage of investigation includes these:

- The software which is running on windows XP operating system can be easily installed.
- The hardware required is Pentium based server.

Operational Feasibility: This feasibility test asks if the system will works when it is developed is installed

Operational feasibilities in this project are:

- The proposed system offers greater level of user-friendly lines.
- The proposed system produces best results and gives high performance.
- It can be implemented easily. So, the project is operational feasible.

Economical feasibility: The economic feasibility step of business development is that period during which a break-even financial model of the business venture is developed based on all costs associated with taking the product from idea to market and achieving sales sufficient to satisfy debt or investment requirements.

Economic feasibility elements include, but are not limited to:

- Increased agency revenue, Decreased agency revenue,
- Increased agency costs, Decreased agency costs,
- Increased revenue to other agencies and/or the general public, Decreased revenue to other agencies and/or the general public,
- Increased costs to other agencies and/or the general public, Decreased costs to other agencies and/or the general public, and,
- Other public benefits

Problem formulation:

The **ONLINE MEDICAL STORE MANAGEMENT SYSTEM** is the specialized system developed after looking over the de-merits of not keeping details of medicine stocks in the shop. Some of them are as follows:

- → **LOSS OF MONEY and MEDICINES:** It is very hard to manage the shops in this big era and due to the same reasons, it can be concluded that this causes a lot of wastage of money as well as the medicines.

  (It is found that three to seven per cent of medications intended for patients go unused inside the United States of America and that similar wastage could value Canada about $8 billion annually.)

- → **DIFFICULTIES FOR THE FACILITIES AT MEDICAL STORE:** As discussed that it is very hard to manage the resources available in the medical store. This causes a lot of trouble not only to the medical store professionals but many more.

- → This also leads to **LOSS OF TIME** while making the records with the hand or managing the registers.

- → While here are the few known demerits of not managing a database are:

  - ✓ REDUNDANCY
  - ✓ UPDATE ERRORS AND REDUCED CONSISTENCY
  - ✓ LESS DATA SECURITY (as anyone can access your registers)

Problem Objectives:

(Objectives proposed before the finalization of project)

→ The proposed work is aimed to carry out work leading to the development of an approach for Online medical store management system. The proposed aim will be achieved by dividing the work into following objectives:

1. To provide searching facilities on the basis of various factors like stocks and theirs.
2. To manage the user details for analyzing sales and expenses.
3. To keep track and manage all information.
4. To show the description and information of the stocks and others.
5. To enhance the efficiency of management in medical store. To monitor the information regarding the transaction of the sales.
6. To improve the function of edit, add and update the records for the proper resource management.

(Current objectives)

→ Along with the above-mentioned objectives, we are on aim to have some more objectives which are as follows:

As we all know that "**Every problem has a solution**". So, here is the solution to all your problems where you can keep records of medicines and this will definitely lead to solve the above-mentioned problems in the following ways:

→ As you will keep the records of medicine, it will help in not wasting the medicines. This will decrease the shortage as well as the wastage of medicines all over the world. Not only this, it will also save your money.

→ It will make the task of professionals easy in keeping the records of the medicine and helps in easy management in the medical store.

METHODOLOGY

Methodology of this is contributed with the help of diagrams which are generally drawn while proposing a project.

So, we have come with some of the following diagrams based upon the systems of our project:

1. Data flow diagrams
2. Use case diagrams
3. Sequence diagrams
4. Deployment diagrams
5. ER diagrams
6. Component diagrams
7. Class diagrams
8. Activity diagrams

We need to draw all these diagrams because of the following reasons:

UML diagrams can be used as a way, to visualize a project before it takes place or as documentation for a project thereafter.

But the broad goal of UML diagrams is to permit teams to visualize how a project is or will be operating, and they can be used in any domain, not just software engineering.

The diagrams will allow crews to visualize together how a system or procedure will work or did work.

It can furnish new ideas for how the team needs to cooperate to achieve the goal of the workflow process.

A UML diagram won't help a team imagine a project and collaborate if a team member is unable to see the diagram and any updates or changes in real-time.

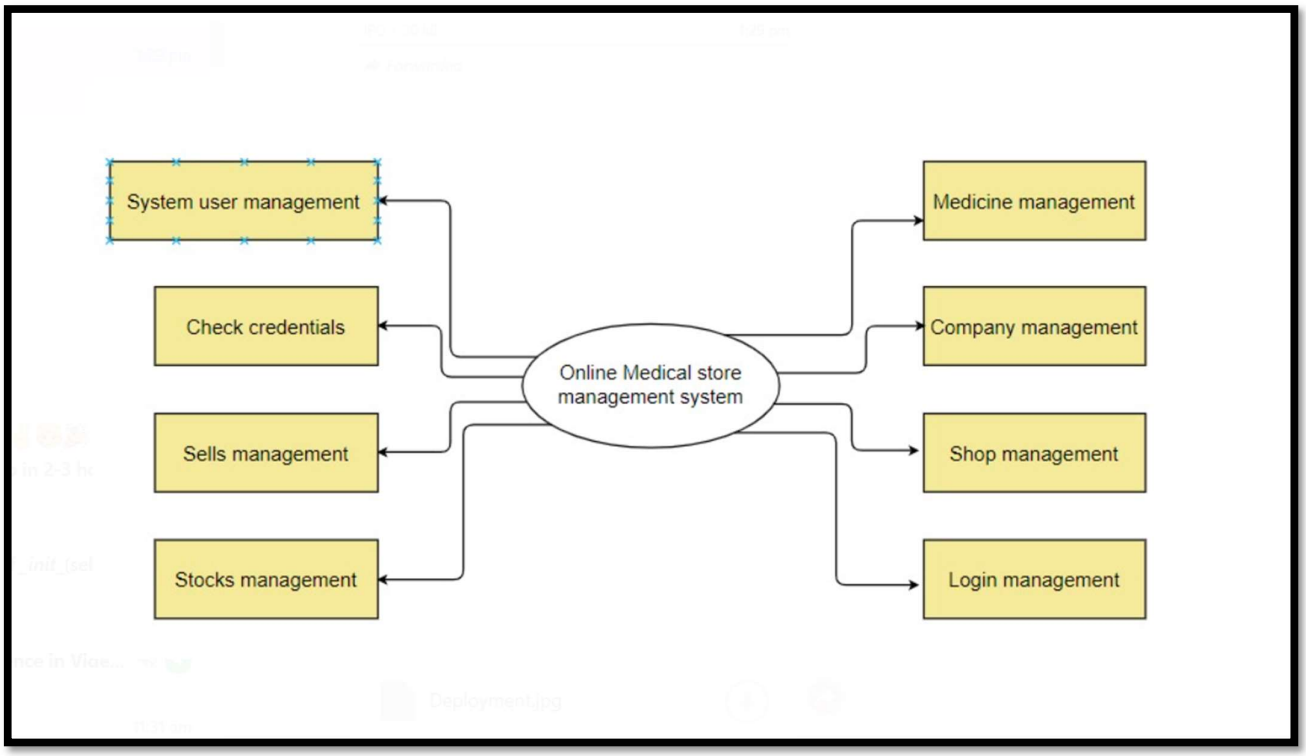
DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system.

A neat and clean DFD can depict the proper quantity of the machine requirement graphically.
It may be manual, automated, or a aggregate of both.

It indicates how statistics enters and leaves the machine, what modifications the information, and wherein statistics is stored.

The goal of a DFD is to expose the scope and obstacles of a machine as a whole.



It can be used as a communique device among a machine analyst and any character who performs a component within side the order that acts as a start line for remodeling a machine.

The DFD is likewise known as a statistics float graph or bubble chart.
DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system.

The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams.

DFD has often been used due to the following reasons:

- Logical information flow of the system
- Determination of physical system construction requirements
- Simplicity of notation
- Establishment of manual and automated systems requirements.

Our project Online Medical Management system data flow diagram is given below:

- Firstly, there is a External Entity (also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed) named Online medical store management system shows a process that transforms data inputs into data output.
- Then, there is a association or a joint or line that connects the Online medical store management system entity to the rest of all the various external entities or the data store boxes where the data is stored.
- The various entities or the boxes where the data stored named as data store boxes (A data store does not generate any operations but simply holds data for later access) are Student user management, check credentials, shop management, login management, company management, medicine management, Sells management, stocks management.

USE CASE DIAGRAMS:

USE CASE DIAGRAM

In the Unified Modeling Language (UML), a use case diagram can summarize the info of your system's users (additionally referred to as actors) and their interactions with the system.

To construct one, you may use a hard and fast of specialized symbols and connectors. It models the tasks, services, and functions required by a system/subsystem of an application.

It depicts the high-level functionality of a system and also tells how the user handles a system. The major motive of a use case diagram is to painting the dynamic issue of a system.

It accumulates the system's requirement, which incorporates each inner in addition to outside influences.

It invokes persons, use cases, and numerous matters that invoke the actors and factors responsible for the implementation of use case diagrams

Following are the purposes of a use case diagram given below:

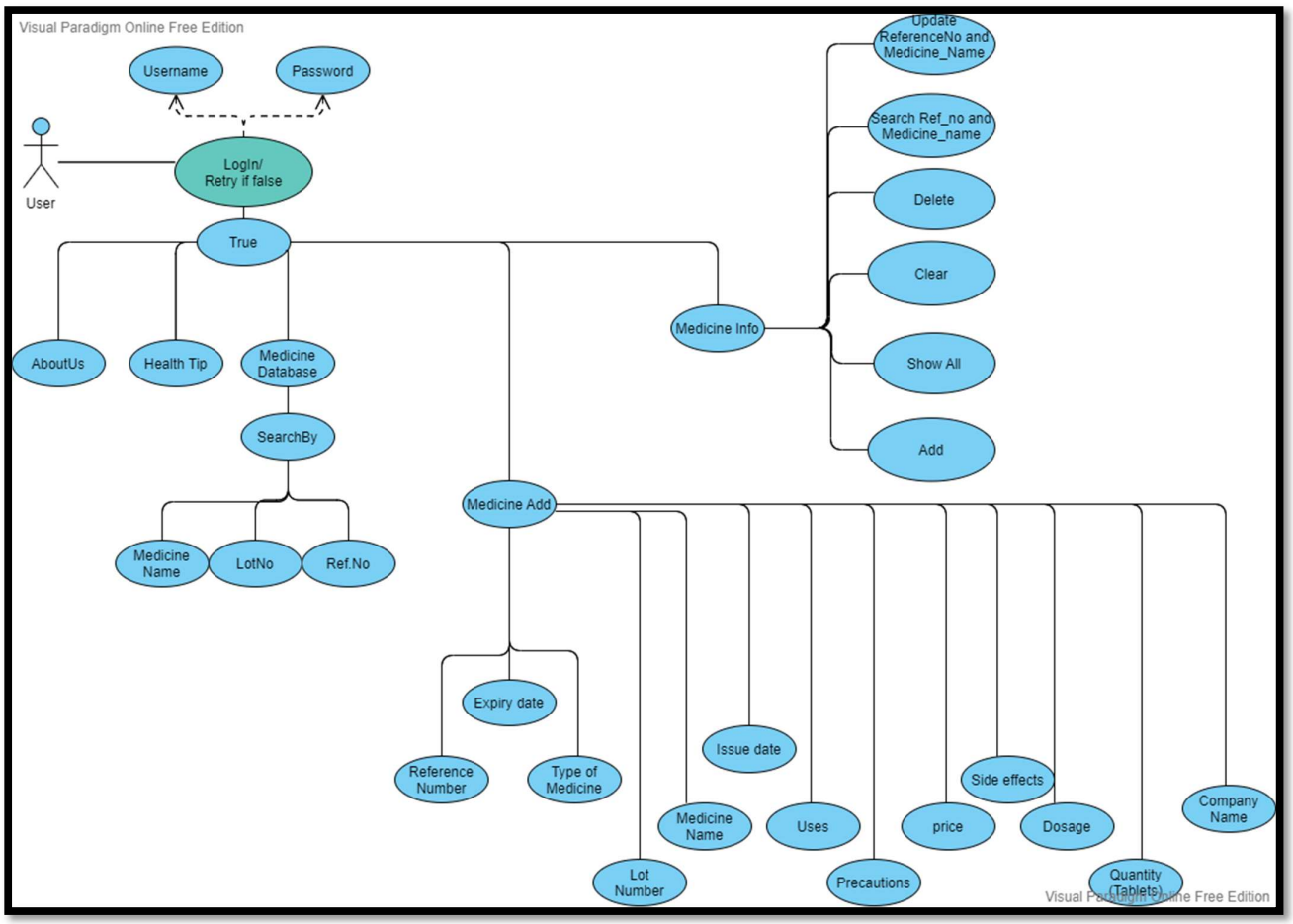
- It gathers the system's needs.
- It depicts the external view of the system.
- It recognizes the internal as well as external factors that influence the system.
- It represents the interaction between the actors.

Use Case Diagram common components include: Actors, System, Associations, System Boundary Boxes, Packages, Goals.

Our project Online Medical Store Management system use case diagram is given below:

- It contains only one Actor named as User.
- It has many use cases in it firstly it has Login/Retry if else named use case which has various attributes in it named as True.
- The communication Link in this connects the system likewise firstly it connect the actor user to the login use case and then connects al the use cases and attributes to each other.

- It has Extend function which is used between login to the username and password.
- The True use case has various attributes in it like About Us, Health Tip, Medicine Database, Medicine Add and Medicine Info. in it.
- The Search By use case has attributes like Medicine Name, Lot No, Ref No.
- The Medicine Info Use case has attributes like Update Reference no and medicine _name, search ref _no and medicine _name, Delete, Clear, Show All and add attribute in it.
- The medicine Add use case has many attributes these are : reference number, expiry date, type of medicine, lot number, medicine name, issue date, uses, precautions, dosage, price, quantity, company name, side effect.



CLASS DIAGRAMS:

Class diagram

The Class diagram depicts a static view of an application. It represents the kinds of items dwelling within side the gadget and the relationships among them. A elegance includes its items, and additionally it is able to inherit from different classes. A elegance diagram is used to visualize, describe, file numerous one of a kind elements of the gadget, and additionally assemble executable software program code.It indicates the attributes, classes, capabilities, and relationships to provide an outline of the software program gadget. It constitutes elegance names, attributes, and capabilities in a separate compartment that facilitates in software program development. Since it's far a group of classes, interfaces, associations, collaborations, and constraints, it's far termed as a structural diagram.

The main purpose of class diagrams is to construct a static view of an application. It is the best diagram this is extensively used for construction, and it is able to be mapped with object-orientated languages. It is one of the maximum famous UML diagrams.

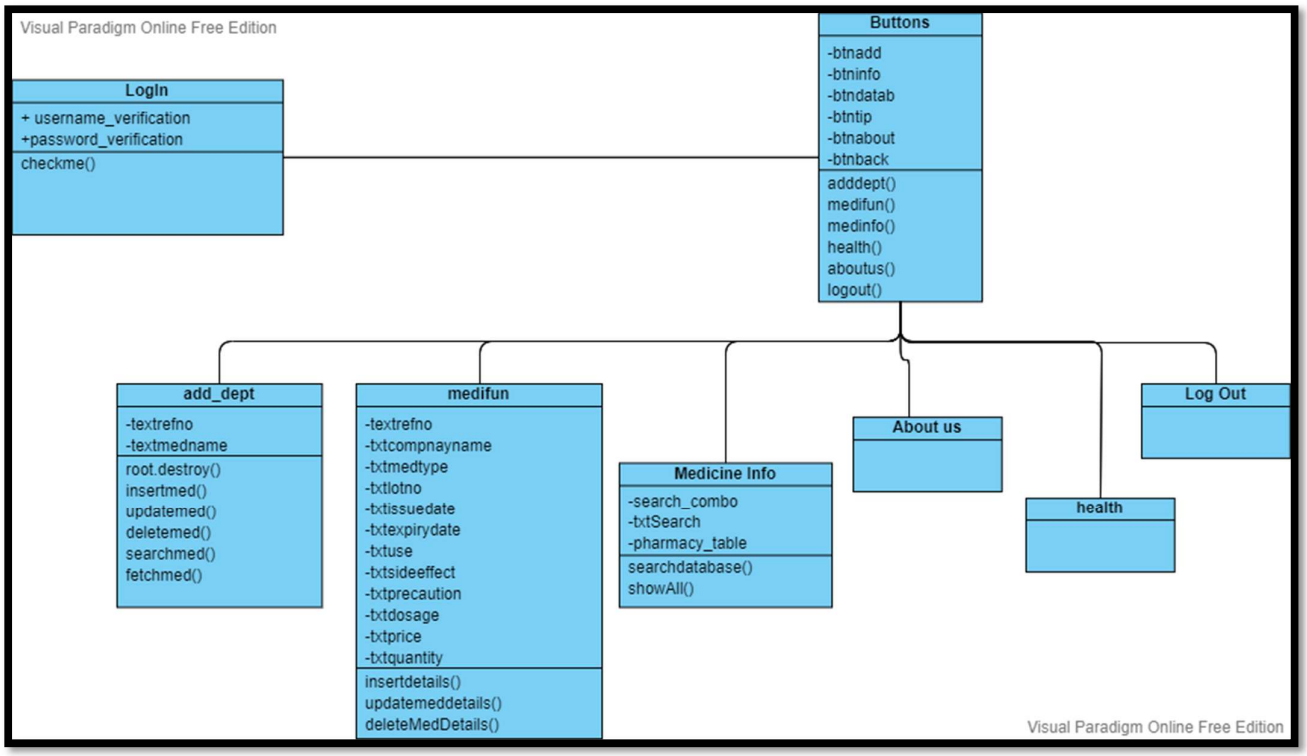
Following are the cause of sophistication diagrams given below:

- It analyses and designs a static view of an application.
- It describes the major responsibilities of a system.
- It is a base for component and deployment diagrams.
- It incorporates forward and reverse engineering.

Our Online Medical Store Management System Project class diagram is here:

- Firstly, it has the graphical representation of the class named as Login class which has two attributes and one operation in it named as check me() operation .
- Then, there is a simple association which joins the two classes Login and the Buttons, the class buttons contain six attributes in it and six operations in it that are add dept(),med fun(),med info(),health(),about us() and logout() .

- Then, there is also association between the class buttons and add _dept, med fun, Medicine info, about us, health and log out classes in it.
- The add _dept class has 2 attributes and 6 operations that are root. destroy(), insert med(),update med(), search med(), fetch med().
- The med fun class has 12 attributes and 3 operations that are insert details(),update med details(),delete med details().
- The Medicine Info class has 3 attributes and 2 operations that are search database(), show All().
- The About Us class has no attributes and no operations in it.
- The health class has no attributes and no operations in it.
- The Log Out class has no attributes and no operations in it



COMPONENT DIAGRAM:

A component diagram is used to interrupt down a massive object-orientated gadget into the smaller additives, for you to cause them to extra manageable. It fashions the bodily view of a gadget including executables, files, libraries, etc. that is living within side the node.

It visualizes the relationships in addition to the employer among the additives gift within side the gadget. It enables in forming an executable gadget. A factor is a unmarried unit of the gadget, that is replaceable and executable.

The implementation info of a factor are hidden, and it necessitates an interface to execute a function. It is sort of a black container whose conduct is defined via way of means of the supplied and required interfaces.

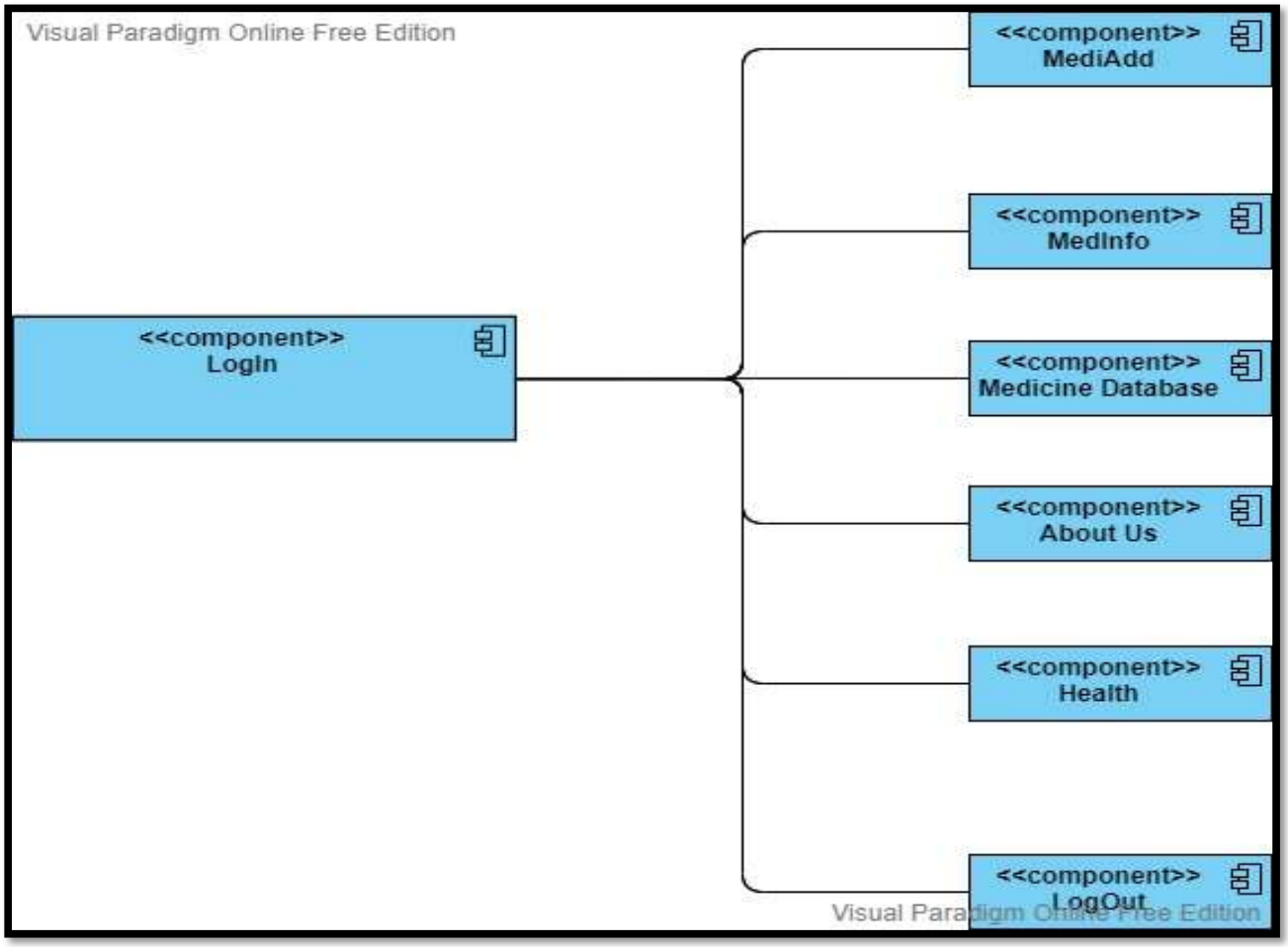
Component diagrams are used to visualize the organization and relationships among components in a system. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system. These diagrams are also used to make executable systems.

The main purpose of the component diagram are enlisted below:

- It envisions each component of a system.
- It constructs the executable by incorporating forward and reverse engineering.
- It depicts the relationships and organization of components.

Our Online Medical Store Management System Project component diagram has:

- Firstly, there is a rectangle with the component icon and the component stereotype named as Login.
- Then, there is a association (association specifies a semantic relationship that can occur between typed instances. It has at least two ends represented by properties, each of which is connected to the type of the end) between component class login and other component
  classes.
- The component classes are Medi Add, Med Info, Medicine Database, About Us, Health, Log Out.



SEQUENCE DIAGRAMS:

Sequence diagram

The sequence diagram represents the waft of messages within side the device and is likewise termed as an occasion diagram. It enables in envisioning numerous dynamic scenarios.

It portrays the communication among any lifelines as a time-ordered series of events, such that those lifelines took component on the run time. In UML, the lifeline is represented via way of means of a vertical bar, while the message flow is represented via way of means of a vertical dotted line that extends throughout the lowest of the page.

One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams. In addition to their use in designing new systems, sequence diagrams can be used to document how objects in an existing system currently interact It consists of the iterations in addition to branching.

Purpose of sequence diagram are:-

- To model high-level interaction among active objects within a system.
- To model interaction among objects inside a collaboration realizing a use case.
- It either models generic interactions or some certain instances of interaction.

The following scenarios are ideal for using a sequence diagram:

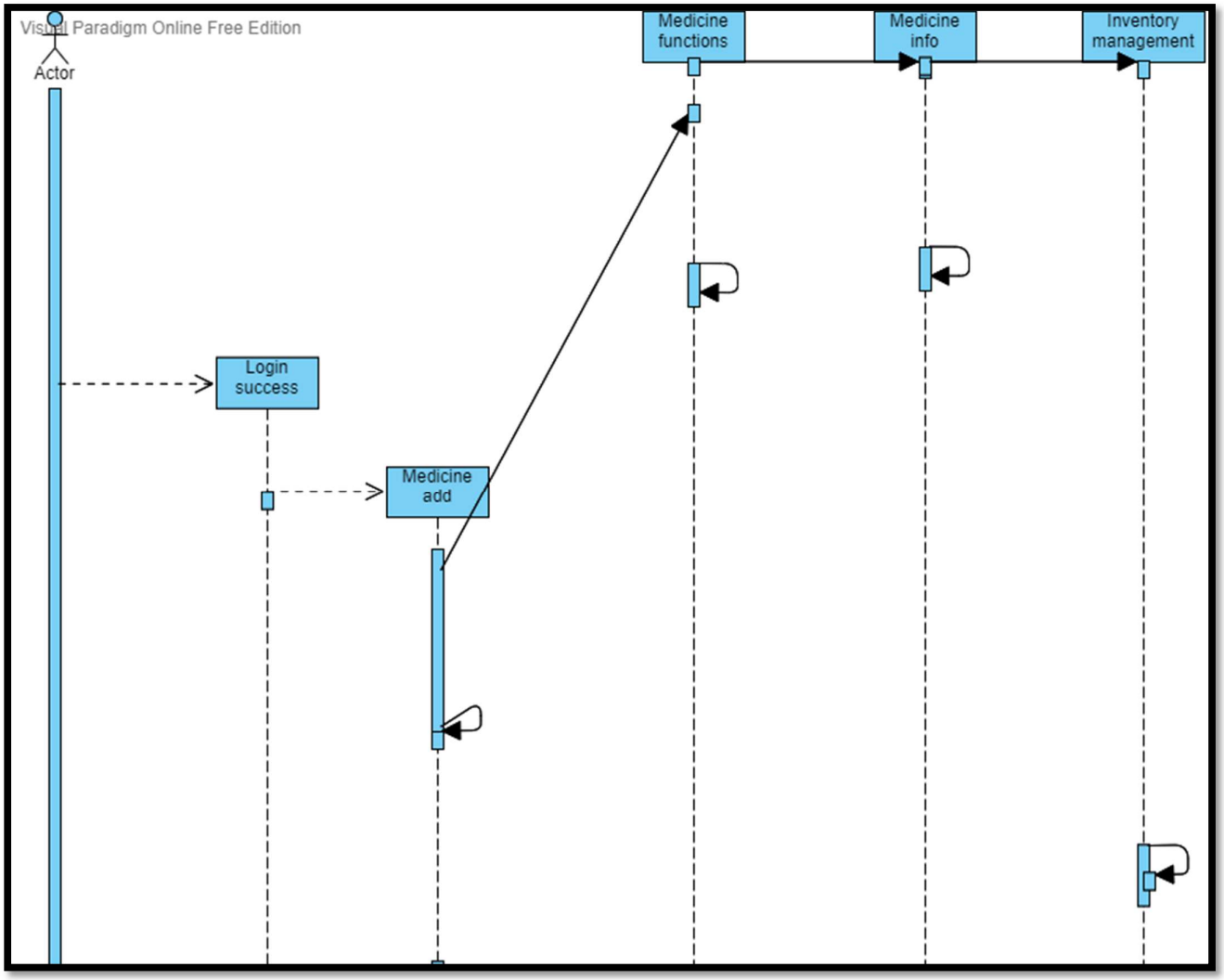
- Usage Scenario
- Service Logic
- Method Logic
- Sequence Diagram Visio

Benefits of a Sequence Diagram:

- It explores the real-time application.
- It depicts the message flow between the different objects.
- It has easy maintenance.
- It is easy to generate.

The drawback of a Sequence Diagram

- In the case of too many lifelines, the sequence diagram can get more complex.
- The incorrect result may be produced, if the order of the flow of messages changes.
- The type of sequence is decided by the type of message.



ACTIVITY DIAGRAMS:

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs.

The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc. It mainly models processes and workflows. It envisions the dynamic behavior of the system as well as constructs a runnable system that incorporates forward and reverse engineering.

It does not include the message part, which means message flow is not represented in an activity diagram. It is the same as that of a flowchart but not exactly a flowchart itself. It is used to depict the flow between several activities. It is also termed as an object-oriented flowchart.

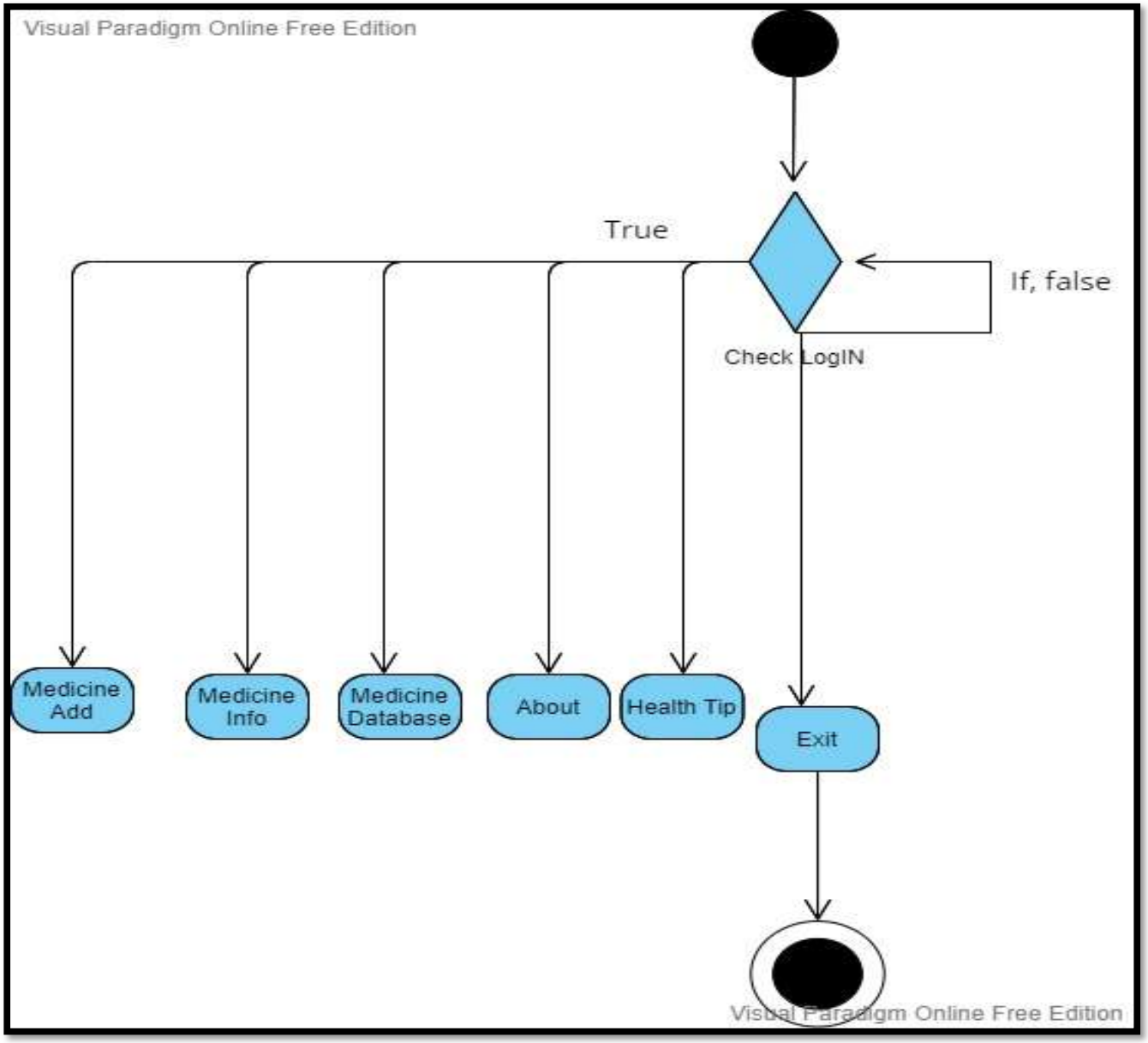
It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.

An activity diagram can be used to portray business processes and workflows. Also, it used for modeling business as well as the software. An activity diagram is utilized for the followings:

- To graphically model the workflow in an easier and understandable way.
- To model the execution flow among several activities.
- To model comprehensive information of a function or an algorithm employed within the system.
- To model the business process and its workflow.
- To envision the dynamic aspect of a system.

Our Online Medical Store Management System Project component diagram has:

- Firstly, there is a initial state or a start point from where this goes into the decision symbol named as check login.
- Then, from the decision login symbol there are 6 activity states in it named as medicine add, medicine info, about, health tip, medicine database, exit.
- In this there are guards which checks that if the login credentials are true then only it goes into the following activity states if its false then it goes into activity sate named as exit.



DEPLOYMENT DIAGRAMS:

The deployment diagram visualizes the bodily hardware on which the software program might be deployed.

It portrays the static deployment view of a device.
It includes the nodes and their relationships.

It ascertains how software program is deployed at the hardware.
It maps the software program structure created in layout to the bodily device structure, wherein the software program might be carried out as a node.

Since it includes many nodes, the connection is proven via way of means of using communique paths.
The main purpose of the deployment diagram is to symbolize how software program is hooked up at the hardware factor.

The deployment diagram is mostly employed by network engineers, system administrators, etc. with the purpose of representing the deployment of software on the hardware system.

It envisions the interaction of the software with the hardware to accomplish the execution.
It depicts in what way a software program interacts with hardware to carry out its execution.

Both the deployment diagram and the factor diagram are carefully interrelated to every different as they awareness on software program and hardware additives.

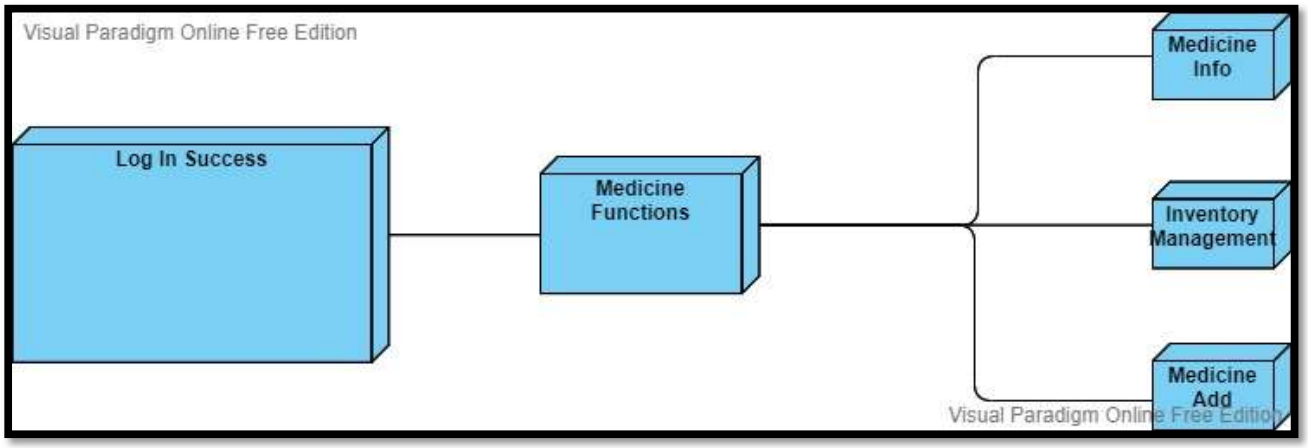
The factor diagram represents the additives of a system, while the deployment diagram describes how they're sincerely deployed at the hardware.

Following are the purposes of deployment diagram enlisted below:

- To envision the hardware topology of the system.
- To represent the hardware components on which the software components are installed.
- To describe the processing of nodes at the runtime.

Our Online Medical Store Management System Project Deployment diagram have:

- Firstly, there is a node (is a deployment target which represents computational resource upon which artifacts may be deployed for execution.) named as Login In success.
- Then, the log in success node connects with the another node through a line, node named as Medicine Functions.
- Then, the medicine functions node connects with the other three nodes named as Medicine Info, Inventory Management and medicine add node.



ER DIAGRAM:

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships. ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships. ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems. Entity relationship diagrams are used in software engineering during the planning stages of the software project. They help to identify different system elements and their relationships with each other. It is often used as the basis for data flow diagrams or DFD's as they are commonly known.

The prime reasons for using the ER Diagram:

- Helps you to define terms related to entity relationship modeling.
- Provide a preview of how all your tables should connect, what fields are going to be on each table.
- Helps to describe entities, attributes, relationships.
- ER diagrams are translatable into relational tables which allows you to build databases quickly.

The components of ER Diagram are entities, attributes , relationships .

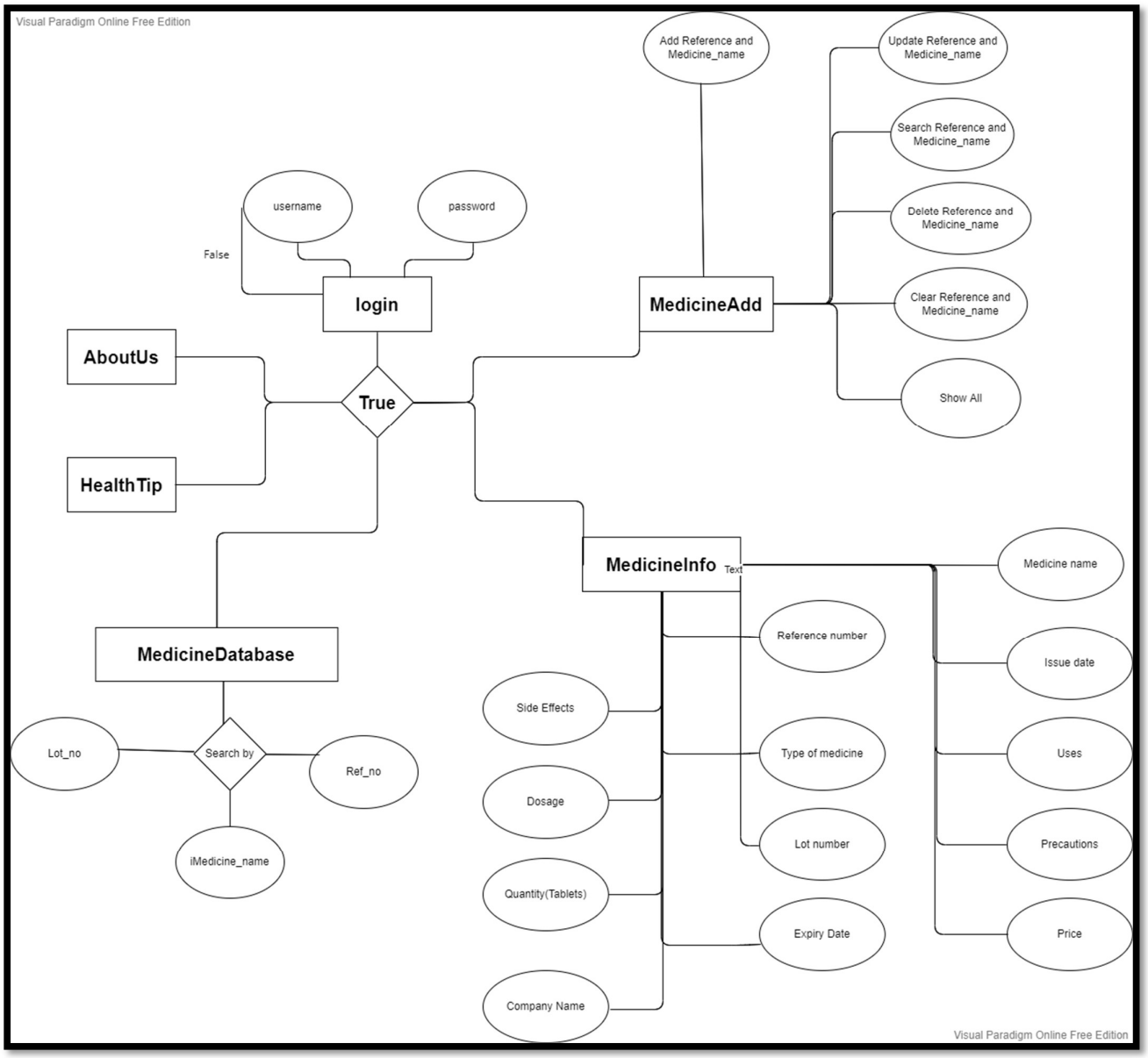
Purpose of ERD:

- The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD.
- The ERD serves as a documentation tool.

- Finally, the ERD is used to connect the logical structure of the database to users. In particular, the ERD effectively communicates the logic of the database to users.

Below points show how to go about creating an ER diagram.

- Identify all the entities in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
- Identify relationships between entities. Connect them using a line and add a diamond in the middle describing the relationship.
- Add attributes for entities. Give meaningful attribute names so they can be understood easily



RESULTS AND DISCUSSIONS

Code for the program:

```
from tkinter import *
from PIL import Image,ImageTk
from tkinter import ttk
import tkinter as pk
import mysql.connector
import random
from tkinter import messagebox
from os import system
import psycopg2
import datetime

connectiondb
=psycopg2.connect(host="127.0.0.1",user="postgres",password="chirag15A",database="pharma
", port = "5432")
cursordb = connectiondb.cursor()

#main frame size
root= Tk()
root.geometry("1920x1000+0+0")

#main heading
lbltitle = Label(root,text = "Online Medical store management system", bd = 15, relief =
RIDGE,
            bg = "white", fg='teal', font = ("Bahnschrift SemiBold",50,"bold"),padx = 2,
pady = 4)
lbltitle.pack(side=TOP,fill=X)

#image for logo
img1 = Image.open("newlogo.jpeg")
img1 = img1.resize((80,90),Image.ANTIALIAS)
photoimg1 = ImageTk.PhotoImage(img1)
b1=Button(root, image=photoimg1,borderwidth=0)
b1.place(x=15,y=15)

#frame for workspace
Frames = Frame(root, bd = 15, relief = FLAT, padx=20, bg="teal")
Frames.place(x=0, y = 120, width = 1920, height = 890)
```

```
def login():
    global username_verification
    global password_verification
    username_verification = StringVar()
    password_verification = StringVar()
    Frames1 = Frame(Frames, bd = 15, relief = RIDGE, padx=20)
    Frames1.place(x=0, y = 100, width = 1450, height = 500)
    lblhead = Label(Frames1,font = ("Arial",48,"bold"), text = "Login here!",  pady =
10)
    lblhead.place(x=550, y=00)
    lbluser = Label(Frames1,font = ("Arial",24,"bold"), text = "Enter username: ",  pady
= 10)
    lbluser.place(x=350, y=100)
    entryuser = Entry(Frames1, textvariable=username_verification, font=('arial', 18,
'bold'), relief = RIDGE)
    entryuser.place(x = 770, y =112)
    lblpass = Label(Frames1,font = ("Arial",24,"bold"), text = "Enter password: ",  pady
= 10)
    lblpass.place(x=350, y=200)
    entrypass = Entry(Frames1, textvariable=password_verification,
show="*",font=('arial', 18, 'bold'), relief = RIDGE)
    entrypass.place(x = 770, y =212)
    b3 = Button(Frames1, text="Login", bg="white", fg='teal', relief="ridge",
font=('arial', 12, 'bold'), width = 20, command = checkme)
    b3.place(x = 600, y =300)
def checkme():
    user_verification = username_verification.get()
    pass_verification = password_verification.get()
    try:
        int(pass_verification)
        sql = "select * from login where username = %s and password = %s"
        cursordb.execute(sql,[(user_verification),(pass_verification)])
        results = cursordb.fetchall()
        if results:
            for i in results:
                logged()
                buttons()
                break
        else:
            failed()
    except ValueError:
```

```
        messagebox.showinfo("Error", "Input Password correctly!")
def logged():
    Frames1 = Frame(Frames, bd = 15, relief = RIDGE)
    Frames1.place(x=0, y = 100, width = 1450, height = 500)
    global logged_message
    logged_message = Toplevel(Frames1)
    logged_message.title("Welcome")
    logged_message.geometry("500x100")
    Label(logged_message, text="Login Successfully!... Welcome {}
".format(username_verification.get()), fg="green", font="bold").pack()
    Label(logged_message, text="").pack()
    health()
    Button(logged_message, text="OK", bg="teal", fg='white', relief="ridge",
font=('arial', 12, 'bold'), command=logged_destroy).pack()
def failed():
    Frames1 = Frame(Frames, bd = 15, relief = RIDGE)
    Frames1.place(x=0, y = 100, width = 1450, height = 500)
    global failed_message
    failed_message = Toplevel(Frames1)
    failed_message.title("Invalid Message")
    failed_message.geometry("500x100")
    Label(failed_message, text="Invalid Username or Password", fg="red",
font="bold").pack()
    Label(failed_message, text="").pack()
    Button(failed_message,text="Ok", bg="teal", fg='white', relief="ridge",
font=('arial', 12, 'bold'), command=failed_destroy).pack()
def logged_destroy():
    logged_message.destroy()
def failed_destroy():
    failed_message.destroy()
    login()
#navigation buttons
def buttons():
    Frames2 = Frame(Frames, bd = 15, relief = FLAT, padx=15, bg ="teal")
    Frames2.place(x=0, y = 10, width = 1450 , height = 65)

    #button to add medicine
    btnadd = Button(Frames2,text="MEDICINE ADD",  font = ("times new
roman",12,"bold"),fg="teal",command= adddept,
                    bg = "white" ,padx = 2, width = 20)
    btnadd.place(y = 0, x = 100)
```

```
    #button to add medicine distributors
    btninfo = Button(Frames2,text="MEDICINE INFO.",  font = ("times new
roman",12,"bold"),fg="teal", command= medifun,
                     bg = "white",padx = 2, width = 20)
    btninfo.place(y = 0, x = 300)

    #button to view medicine database
    btndatab = Button(Frames2,text="MEDICINE DATABASE",  font = ("times new
roman",12,"bold"),fg="teal", command= medinfo,
                     bg = "white",padx = 2, width = 20)
    btndatab.place(y = 0, x = 500)

    #button to view daily health tip
    btntip = Button(Frames2,text="DAILY HEALTH TIP",  font = ("times new
roman",12,"bold"),fg="teal", command= health,
                     bg = "white",padx = 2, width = 20)
    btntip.place(y = 0, x = 700)

    #button tp go back to know about functions of our buttons
    btnabout = Button(Frames2,text="ABOUT US",  font = ("times new
roman",12,"bold"),fg="teal", command = aboutus,
                     bg = "white",padx = 2, width = 20)
    btnabout.place(y = 0, x = 900)

    #button to exit our code
    btnback = Button(Frames2,text="EXIT",  font = ("times new
roman",12,"bold"),fg="teal", command = logout,
                     bg = "white",padx = 2, width = 20)
    btnback.place(y = 0, x = 1100)
#View health tip
def health():
    #navigation buttons
    buttons()
    #small main frame
    Frames1 = Frame(Frames, bd = 15, relief = RIDGE, padx=20)
    Frames1.place(x=0, y = 100, width = 1450, height = 500)
    #heading
    lblhead = Label(Frames1,font = ("Arial",48,"bold"), text = "Today's Health tip for
you!!",  pady = 10)
    lblhead.place(x=280, y=100)
```

```
    #random function to allot health tip
    list1 = ["Eat a healthy diet", "Consume less salt and sugar", "Reduce intake of
harmful fats", "Avoid harmful use of alcohol"
                ,"Don’t smoke", "Be active", "Check your blood pressure regularly", "Get
tested", "Get vaccinated"
                ,"Prevent mosquito bites","Drink only safe water","Take antibiotics only
as prescribed","Clean your hands properly","Have regular check-ups"]
    lblrefno = Label(Frames1,font = ("times new roman",24,"bold"), text =
random.choice(list1),  pady = 10)
    lblrefno.place(relx=0.5, rely=0.5, anchor=CENTER)
#to insert medicine:
def insertmed():
    m = reference_number.get()
    try:
        int(m)
        cursordb.execute("SELECT * FROM medicall WHERE refno = %s;", (
reference_number.get(),))
        connectiondb.commit()
        if cursordb.fetchall():
            messagebox.showinfo("Error", "Reference number already exists!")
        else:
            cursordb.execute("INSERT INTO medicall VALUES(%s, %s)",
(reference_number.get(), medicine_name.get(),))
            connectiondb.commit()
            messagebox.showinfo("Success", "Medicine added!")
            adddept()
    except ValueError:
        messagebox.showinfo("Error", "Input reference number correctly!")
#to update medicine
def updatemed():
    m = reference_number.get()
    try:
        int(m)
        cursordb.execute("SELECT * FROM medicall WHERE refno = %s;",
(reference_number.get(),))
        connectiondb.commit()
        if len(cursordb.fetchall()) == 0:
            messagebox.showinfo("Error", "No reference found!")
            adddept()
        else:
            cursordb.execute("UPDATE medicall SET mediname = %s WHERE refno = %s;", (
```

```
medicine_name.get(),reference_number.get(),))
            connectiondb.commit()
            adddept()
            messagebox.showinfo("Success", "Medicine name updated!")
    except ValueError:
        messagebox.showinfo("Error", "Input reference number correctly!")
#to delete database
def deletemed():
    m = reference_number.get()
    try:
        int(m)
        cursordb.execute("SELECT * FROM medicall WHERE refno = %s;",
(reference_number.get(),))
        connectiondb.commit()
        if len(cursordb.fetchall()) == 0:
            messagebox.showinfo("Error", "No reference found!")
            adddept()
        else:
            cursordb.execute("DELETE FROM medicall WHERE refno = %s;",(
reference_number.get(),))
            connectiondb.commit()
            messagebox.showinfo("Success", "Medicine removed!")
            adddept()
    except ValueError:
        messagebox.showinfo("Error", "Input reference number correctly!")
#to search medicine
def searchmed():
    messagebox.askokcancel("askokcancel", "Want to continue?")
    cursordb.execute("SELECT * FROM medicall WHERE mediname = %s OR refno = %s;", (
medicine_name.get(),reference_number.get(),))
    connectiondb.commit()
    rows = cursordb.fetchall()
    adddept()
    for row in rows:
        medicine_table.insert("", pk.END, values=row)
#to show table
def fetchaddmed():
    adddept()
    cursordb.execute("SELECT * FROM medicall")
    rows = cursordb.fetchall()
    for row in rows:
```

```
        medicine_table.insert("", pk.END, values=row)
#Add medicine new in the market
def adddept():
    global reference_number
    global medicine_name
    global medicine_table
    reference_number = StringVar()
    medicine_name = StringVar()
    #navigation buttons
    buttons()
    Frames1 = Frame(Frames, bd = 15, relief = RIDGE, padx=20)
    Frames1.place(x=0, y = 100, width = 1450, height = 500)
    # headings
    lblhead = Label(Frames1,font = ("Arial",24,"bold"), text = "Medicine Add
Department",  pady = 10)
    lblhead.place(x=500, y=00)

    lblrefno = Label(Frames1,font = ("times new roman",20,"bold"), text = "Reference
no.:",  pady = 10)
    lblrefno.place(x=290, y=60)

    #data to be enetered in reference number
    txtrefno = Entry(Frames1,textvariable = reference_number,
bd=2,relief=RIDGE,width=20,font = ("times new roman",21,"bold"))
    txtrefno.place(x=790, y=75)

    lblmedname = Label(Frames1,font = ("times new roman",20,"bold"), text = "Medicine
name:", padx = 2,  pady = 10)
    lblmedname.place(x=290, y=140)

    #data to be entered in medicine name
    txtmedname = Entry(Frames1, textvariable=
medicine_name,bd=2,relief=RIDGE,width=20,font = ("times new roman",21,"bold"))
    txtmedname.place(x=790, y=150)

    #graph where medicine name and referene number will be displayed
    side_frame = Frame(Frames1, bd=4, relief=RIDGE, bg="white")
    side_frame.place(x=25,y=240,width=1330, height=150)

    sc_x=ttk.Scrollbar(side_frame,orient =  HORIZONTAL)
    sc_x.pack(side=BOTTOM, fill=X)
```

```
    sc_y=ttk.Scrollbar(side_frame,orient =  VERTICAL)
    sc_y.pack(side=RIGHT, fill=Y)
    medicine_table = ttk.Treeview(side_frame, column=("ref",
"medname"),xscrollcommand=sc_x.set,yscrollcommand=sc_y.set)
    sc_x.config(command = medicine_table.xview)
    sc_y.config(command = medicine_table.yview)

    #headings of the graph
    medicine_table.heading('ref', text = 'Reference no')
    medicine_table.heading('medname', text = ' Medicine name')
    medicine_table["show"]= "headings"
    medicine_table.pack(fill=BOTH, expand = 1)

    medicine_table.column('ref', width = 100)
    medicine_table.column('medname', width = 100)


    #medicine add buttons
    down_frame=Frame(Frames1,bd=4,relief=FLAT)
    down_frame.place(x=25,y=400, width=1330, height = 40)

    #button to add / update / delete / clear  medicine
    btnAdd = Button(down_frame,text="ADD",  font = ("times new
roman",12,"bold"),fg="teal", command = insertmed,
                          bg = "white",padx = 2, width = 13 )
    btnAdd.place(x=100,y=0)
    btnupd = Button(down_frame,text="UPDATE",  font = ("times new
roman",12,"bold"),fg="teal", command = updatemed,
                          bg = "white",padx = 2, width = 13)
    btnupd.place(x=300,y=0)
    btndel = Button(down_frame,text="DELETE",  font = ("times new
roman",12,"bold"),fg="teal", command = deletemed,
                          bg = "white",padx = 2, width = 13)
    btndel.place(x=500,y=0)
    btnsearch = Button(down_frame,text="SEARCH",  font = ("times new
roman",12,"bold"),fg="teal", command = searchmed,
                          bg = "white",padx = 2, width = 13)
    btnsearch.place(x=700,y=0)
    btnclear = Button(down_frame,text="CLEAR",  font = ("times new
roman",12,"bold"),fg="teal", command = adddept,
                          bg = "white",padx = 2, width = 13)
    btnclear.place(x=900,y=0)
```

```
    btnAdd = Button(down_frame,text="SHOW ALL",  font = ("times new
roman",12,"bold"),fg="teal", command = fetchaddmed,
                          bg = "white",padx = 2, width = 13 )
    btnAdd.place(x=1100,y=0)
#search function:
def searchdatabase():
    val = search_by.get()
    if val == "Reference number":
        try:
            int(findme.get())
            messagebox.askokcancel("askokcancel", "Want to continue?")
            cursordb.execute("SELECT * FROM medidetails WHERE refno = %s",
(findme.get(),))
            rows = cursordb.fetchall()
            if len(rows) == 0:
                messagebox.showinfo("Error", "Reference number not found!")
                medinfo()
            else:
                medinfo()
                for row in rows:
                    pharmacy_table.insert("", index = 0, values=row)
        except ValueError:
            messagebox.showinfo("Error", "Input reference number correctly!")
    elif val == "Medicine name":
        if findme.get() == "":
            messagebox.showinfo("Error", "Please input value!")
        else:
            messagebox.askokcancel("askokcancel", "Want to continue?")
            cursordb.execute("SELECT * FROM medidetails WHERE mediname = %s",
(findme.get(),))
            rows = cursordb.fetchall()
            if len(rows) == 0:
                messagebox.showinfo("Error", "Medicine name not found!")
                medinfo()
            else:
                medinfo()
                for row in rows:
                    pharmacy_table.insert("", index = 0, values=row)
    else:
        try:
            int(findme.get())
```

```
            messagebox.askokcancel("askokcancel", "Want to continue?")
            cursordb.execute("SELECT * FROM medidetails WHERE lotno = %s",
(findme.get(),))
            rows = cursordb.fetchall()
            if len(rows) == 0:
                messagebox.showinfo("Error", "Lot number not found!")
                medinfo()
            else:
                medinfo()
                for row in rows:
                    pharmacy_table.insert("", index = 0, values=row)
        except ValueError:
            messagebox.showinfo("Error", "Input Lot number correctly!")
#show full database
def showall():
    medinfo()
    cursordb.execute("SELECT * FROM medidetails")
    rows = cursordb.fetchall()
    for row in rows:
        pharmacy_table.insert("", pk.END, values=row)
#medicine functions database
def medinfo():
    global search_by
    global findme
    global pharmacy_table
    findme = StringVar()
    search_by = StringVar()
    #navigation button
    buttons()
    #main small frame
    Frames1 = Frame(Frames, bd = 15, relief = RIDGE, padx=20)
    Frames1.place(x=0, y = 100, width = 1450, height = 500)
    #main headings
    lblhead = Label(Frames1,font = ("Arial",24,"bold"), text = "Medicine
Information",  pady = 10)
    lblhead.place(x=525, y=0)

    lblhead = Label(Frames1,font = ("Arial",20,"bold"), text = "     Here is the
database of all the medicine that is available in your store",  pady = 10)
    lblhead.place(x=200, y=50)
    #Label - Search by
```

```
    lblsearch = Label(Frames1,font = ("arial",14,"bold"), text = "Search by", padx = 2,
bg="teal", fg="white" , width = 17)
    lblsearch.place(x = 20, y = 100)
    #Options to choose on what basis do you want to search
    search_combo= ttk.Combobox(Frames1, textvariable = search_by,width = 17, font =
("times new roman",14,"bold"),state="readonly")
    search_combo["values"]=("Reference number", "Medicine name", "Lot number")
    search_combo.place(x = 300, y = 100)
    search_combo.current(0)
    #Enter your opinion abt what to search
    txtSearch = Entry(Frames1, textvariable = findme, relief=RIDGE ,width=20,font =
("times new roman",16,"bold"))
    txtSearch.place(x = 570, y = 100)
    #click search btn to search
    searchbtn = Button(Frames1,text="SEARCH",  font = ("times new
roman",12,"bold"),fg="teal", command = searchdatabase,
                        bg = "white",padx = 2, width = 17)
    searchbtn.place(x =1000, y= 100)
    #click btn to show all database
    showbtn = Button(Frames1,text="SHOW ALL",  font = ("times new
roman",12,"bold"),fg="teal", command = showall,
                        bg = "white",padx = 2, width = 17)
    showbtn.place(x = 1200, y = 100)

    #Frame Details

    #Maintable and scroll bars - to show database
    table_frame=Frame(Frames1,bd=15,relief=RIDGE, padx = 10, pady = 10)
    table_frame.place(x=0,y=150, width=1380, height = 310)

    sc_xx=ttk.Scrollbar(table_frame,orient =  HORIZONTAL)
    sc_xx.pack(side=BOTTOM, fill=X)
    sc_yy=ttk.Scrollbar(table_frame,orient =  VERTICAL)
    sc_yy.pack(side=RIGHT, fill=Y)

    pharmacy_table=ttk.Treeview(table_frame,
column=("refno","medname","companyname","type","lotno","issuedate","expdate"
        ,"uses","sideeffect","warning","dsage","price","productqt")
        ,xscrollcommand=sc_xx.set,yscrollcommand=sc_yy.set)

    sc_xx.pack(side=BOTTOM, fill=X)
```

```
    sc_yy.pack(side=RIGHT, fill=Y)

    sc_xx.config(command = pharmacy_table.xview)
    sc_yy.config(command = pharmacy_table.yview)

    pharmacy_table["show"]= "headings"

    pharmacy_table.heading('refno', text = 'Reference no')
    pharmacy_table.heading('medname', text = 'Medicine name')
    pharmacy_table.heading('companyname', text = 'Company name')
    pharmacy_table.heading('type', text = 'Medicine type')
    pharmacy_table.heading('lotno', text = 'Lot no')
    pharmacy_table.heading('issuedate', text = 'Issue date')
    pharmacy_table.heading('expdate', text = 'Exp date')
    pharmacy_table.heading('uses', text = 'Uses')
    pharmacy_table.heading('sideeffect', text = 'Side effects')
    pharmacy_table.heading('warning', text = 'Precautions')
    pharmacy_table.heading('dsage', text = 'Dosage')
    pharmacy_table.heading('price', text = 'Price')
    pharmacy_table.heading('productqt', text = 'Quantity')
    pharmacy_table.pack(fill=BOTH,expand=1)

    pharmacy_table.column('refno', width = 100)
    pharmacy_table.column('companyname', width = 100)
    pharmacy_table.column('type', width = 100)
    pharmacy_table.column('medname', width = 100)
    pharmacy_table.column('lotno', width = 100)
    pharmacy_table.column('issuedate', width = 100)
    pharmacy_table.column('expdate', width = 100)
    pharmacy_table.column('sideeffect', width = 100)
    pharmacy_table.column('warning', width = 100)
    pharmacy_table.column('dsage', width = 100)
    pharmacy_table.column('price', width = 100)
    pharmacy_table.column('productqt', width = 100)
#to insert medicine:
def insertmeddetails():
    format = "%Y-%m-%d"
    m = reference_number.get()
    try:
        int(m)
        try:
```

```
            int(lot_number.get())
            try:
                int(med_price.get())
                try:
                    datetime.datetime.strptime(issue_date.get(), format)
                    try:
                        datetime.datetime.strptime(expiry_date.get(), format)
                        cursordb.execute("SELECT * FROM medicall WHERE refno = %s;", (
reference_number.get(),))
                        connectiondb.commit()
                        if len(cursordb.fetchall()) != 0:
                            messagebox.showinfo("Error", "Reference number already
exists!")
                        else:
                            cursordb.execute("INSERT INTO medidetails VALUES(%s,%s, %s,
%s, %s,%s, %s,%s, %s, %s, %s,%s, %s)",
                                                                        (reference_number.ge
t(),
                                                                        medicine_name.get(),
                                                                        company_name.get(),
                                                                        medicine_type.get(),
                                                                        lot_number.get(),
                                                                        issue_date.get(),
                                                                        expiry_date.get(),
                                                                        medicine_uses.get(),
                                                                        side_effects.get(),
                                                                        med_precautions.get(
),
                                                                        med_dosage.get(),
                                                                        med_price.get(),
                                                                        med_quantity.get(),)
)
                            connectiondb.commit()
                            messagebox.showinfo("Success", "Medicine details added!")
                            medifun()
                    except ValueError:
                        messagebox.showinfo("Error", "Input Expiry date in yy - mm - dd
format!")
                except ValueError:
                    messagebox.showinfo("Error", "Input Issue date in yy - mm - dd
format!")
```

```
            except ValueError:
                messagebox.showinfo("Error", "Input Price correctly!")
        except ValueError:
            messagebox.showinfo("Error", "Input Lot number correctly!")
    except ValueError:
        messagebox.showinfo("Error", "Input reference number correctly!")
#to update medicine
def updatemeddetails():
    format = "%Y-%m-%d"
    m = reference_number.get()
    try:
        int(m)
        try:
            int(lot_number.get())
            try:
                int(med_price.get())
                try:
                    datetime.datetime.strptime(issue_date.get(), format)
                    try:
                        datetime.datetime.strptime(expiry_date.get(), format)
                        cursordb.execute("SELECT * FROM medidetails WHERE refno = %s;",
( reference_number.get(),))
                        connectiondb.commit()
                        if len(cursordb.fetchall()) == 0:
                            messagebox.showinfo("Error", "Reference number not found!")
                        else:
                            cursordb.execute("UPDATE medidetails SET refno = %s,
mediname = %s, companyname = %s, meditype = %s, lotno= %s, issuedate = %s, expdate = %s,
uses = %s, sideeff = %s, precautions = %s, dosage = %s, price = %s, quantity = %s WHERE
refno = %s;",
                                                                    (reference_number.ge
t(),
                                                                    medicine_name.get(),
                                                                    company_name.get(),
                                                                    medicine_type.get(),
                                                                    lot_number.get(),
                                                                    issue_date.get(),
                                                                    expiry_date.get(),
                                                                    medicine_uses.get(),
                                                                    side_effects.get(),
                                                                    med_precautions.get(
```

```
),
                                                                    med_dosage.get(),
                                                                    med_price.get(),
                                                                    med_quantity.get(),
                                                                    reference_number.get
(),))
                            connectiondb.commit()
                            messagebox.showinfo("Success", "Medicine details updated!")
                            medifun()
                        except ValueError:
                            messagebox.showinfo("Error", "Input Expiry date in yy - mm - dd
format!")
                    except ValueError:
                        messagebox.showinfo("Error", "Input Issue date in yy - mm - dd
format!")
                except ValueError:
                    messagebox.showinfo("Error", "Input Price correctly!")
            except ValueError:
                messagebox.showinfo("Error", "Input Lot number correctly!")
        except ValueError:
            messagebox.showinfo("Error", "Input reference number correctly!")
#to search medicine
def delmeddetails():
    m = reference_number.get()
    try:
        int(m)
        messagebox.askokcancel("askokcancel", "Want to continue?")
        cursordb.execute("SELECT * FROM medidetails WHERE refno = %s;",
(reference_number.get(),))
        connectiondb.commit()
        if len(cursordb.fetchall()) == 0:
            messagebox.showinfo("Error", "No reference found!")
            adddept()
        else:
            cursordb.execute("DELETE FROM medidetails WHERE refno = %s;",(
reference_number.get(),))
            connectiondb.commit()
            messagebox.showinfo("Success", "Medicine removed!")
            adddept()
    except ValueError:
        messagebox.showinfo("Error", "Input reference number correctly!")
```

```
#Adding medicine distributors
def medifun():
    global reference_number
    global medicine_name
    global  company_name
    global  medicine_type
    global  lot_number
    global  issue_date
    global  expiry_date
    global  medicine_uses
    global  side_effects
    global  med_precautions
    global  med_dosage
    global  med_price
    global  med_quantity
    reference_number = StringVar()
    medicine_name = StringVar()
    company_name = StringVar()
    medicine_type = StringVar()
    lot_number = StringVar()
    issue_date = StringVar()
    expiry_date = StringVar()
    medicine_uses = StringVar()
    side_effects = StringVar()
    med_precautions = StringVar()
    med_dosage = StringVar()
    med_price = StringVar()
    med_quantity = StringVar()
    #navigation buttons
    buttons()
    #main small frame
    Frames1 = Frame(Frames, bd = 15, relief = RIDGE, padx=20)
    Frames1.place(x=0, y = 100, width = 1450, height = 500)
    #headings
    lblhead = Label(Frames1,font = ("Arial",36,"bold"), text = "Medicine data
modifier",  pady = 10)
    lblhead.place(x=450, y=0)
    #Reference nummber:
    lblrefno = Label(Frames1,font = ("times new roman",14,"bold"), text = "Reference
number:", padx = 10, pady = 5, width = 17)
    lblrefno.place (x =0, y = 70)
```

```
    txtrefno= Entry(Frames1, textvariable = reference_number,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtrefno.place (x =300, y = 70)
    #Company name:
    lblcompanyname = Label(Frames1,font = ("times new roman",14,"bold"), text = "Company
name:", padx = 10,  pady = 5,width = 17)
    lblcompanyname.place (x =700, y = 70)
    txtcompanyname = Entry(Frames1, textvariable = company_name,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtcompanyname.place (x =1000, y = 70)
    #Type of medicine
    lblmedtype = Label(Frames1,font = ("times new roman",14,"bold"), text = "Type of
medicine:", padx = 10, pady = 5,width = 17)
    lblmedtype.place (x =0, y = 110)
    medtype_combo= ttk.Combobox(Frames1, textvariable = medicine_type,  width = 30, font
= ("times new roman",14,"bold"),state="readonly")
    medtype_combo["values"]=("Tablet", "Liquid", "Capsules","Topical Medicines","Drops",
"Inhales", "Injections")
    medtype_combo.place (x =300, y = 110)
    #Medicine name
    lblmedname = Label(Frames1,font = ("times new roman",14,"bold"), text = "Medicine
name:  ", padx = 10, pady = 5, width = 17)
    lblmedname.place (x =700, y = 110)
    txtmedname= Entry(Frames1, textvariable = medicine_name,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtmedname.place (x =1000, y = 110)
    #Lot number:
    lbllotno = Label(Frames1,font = ("times new roman",14,"bold"), text = "Lot number:",
padx = 10,  pady = 5,width = 17)
    lbllotno.place (x =0, y = 150)
    txtlotno = Entry(Frames1, textvariable = lot_number, bd=2,relief=RIDGE,width=32,font
= ("times new roman",14,"bold"))
    txtlotno.place (x =300, y = 150)
    #Issue date:
    lblissuedate = Label(Frames1,font = ("times new roman",14,"bold"), text = "Issue
Date:", padx = 10,  pady = 5,width = 17)
    lblissuedate.place (x =700, y = 150)
    txtissuedate = Entry(Frames1, textvariable = issue_date,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtissuedate.place (x =1000, y = 150)
    #Expiry date
```

```
    lblexpdate = Label(Frames1,font = ("times new roman",14,"bold"), text = "Expiry
date:", padx = 10,  pady = 5,width = 17)
    lblexpdate.place (x =0, y = 190)
    txtexpdate = Entry(Frames1, textvariable = expiry_date,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtexpdate.place (x =300, y = 190)
    #Uses:
    lbluse = Label(Frames1,font = ("times new roman",14,"bold"), text = "Uses:", padx =
10,  pady = 5,width = 17)
    lbluse.place (x =700, y = 190)
    txtuse = Entry(Frames1, textvariable = medicine_uses,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtuse.place (x =1000, y = 190)
    #Side effects
    lblsideeffect = Label(Frames1,font = ("times new roman",14,"bold"), text = "Side
effects:", padx = 10,  pady = 5,width = 17)
    lblsideeffect.place (x =0, y =230)
    txtsideeffect = Entry(Frames1, textvariable = side_effects,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtsideeffect.place (x =300, y = 230)
    #Precautions
    lblprecaution = Label(Frames1,font = ("times new roman",14,"bold"), text =
"Precautions:", padx = 10,  pady = 5,width = 17)
    lblprecaution.place (x =700, y = 230)
    txtprecaution = Entry(Frames1, textvariable = med_precautions,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtprecaution.place (x =1000, y = 230)
    #Dosage
    lbldose = Label(Frames1,font = ("times new roman",14,"bold"), text = "Dosage:", padx
= 2,  pady = 10,width = 17)
    lbldose.place (x =0, y = 270)
    txtdose = Entry(Frames1, textvariable = med_dosage, bd=2,relief=RIDGE,width=32,font
= ("times new roman",14,"bold"))
    txtdose.place (x =300, y = 270)
    #Price per tablet
    lblprice = Label(Frames1,font = ("times new roman",14,"bold"), text = "Price:", padx
= 2,  pady = 10,width = 17)
    lblprice.place (x =700, y = 270)
    txtprice = Entry(Frames1, textvariable = med_price, bd=2,relief=RIDGE,width=32,font
= ("times new roman",14,"bold"))
    txtprice.place (x =1000, y = 270)
```

```
    #Number of tablets available
    lblQuantity = Lab`el(Frames1,font = ("times new roman",14,"bold"), text =
"Quantity(Tablets):", padx = 10,  pady = 5,width = 17)
    lblQuantity.place (x =0, y = 310)
    txtQuantity = Entry(Frames1, textvariable = med_quantity,
bd=2,relief=RIDGE,width=32,font = ("times new roman",14,"bold"))
    txtQuantity.place (x =300, y = 310)

    #frame where buttons are made
    down_frame=Frame(Frames1,bd=4,relief=FLAT)
    down_frame.place(x=0,y=350, width=1375, height = 40)

    btnAdd = Button(down_frame,text="ADD",  font = ("times new
roman",12,"bold"),fg="teal", command = insertmeddetails,
                          bg = "white",padx = 2, width = 13 )
    btnAdd.place(x=350,y=0)
    btnupd = Button(down_frame,text="UPDATE",  font = ("times new
roman",12,"bold"),fg="teal", command = updatemeddetails,
                          bg = "white",padx = 2, width = 13)
    btnupd.place(x=550,y=0)
    btndel = Button(down_frame,text="DELETE",  font = ("times new
roman",12,"bold"),fg="teal", command = delmeddetails,
                          bg = "white",padx = 2, width = 13)
    btndel.place(x=750,y=0)
    btnclear = Button(down_frame,text="CLEAR",  font = ("times new
roman",12,"bold"),fg="teal", command = medifun,
                          bg = "white",padx = 2, width = 13)
    btnclear.place(x=950,y=0)
#about us
def aboutus():
    buttons()
    #main small frame
    Frames1 = Frame(Frames, bd = 15, relief = RIDGE, padx=20)
    Frames1.place(x=0, y = 100, width = 1450, height = 500)
    #main headings
    lblhead = Label(Frames1,font = ("Arial",24,"bold"), text = "About us",  pady = 10)
    lblhead.place(x=625, y=0)

    lblhead = Label(Frames1,font = ("Arial",20,"bold"), text = "     Here, you will get
all the necessary information deals with data.",  pady = 10)
    lblhead.place(x=250, y=50)
```

```
    about_frame=Frame(Frames1,bd=15,relief=RIDGE, padx = 10, pady = 10, bg = "white")
    about_frame.place(x=0,y=100, width=1380, height = 350)

    scrollbar = Scrollbar(about_frame)
    scrollbar.pack( side = RIGHT, fill = Y )
    mylist = Listbox(about_frame, yscrollcommand = scrollbar.set, height = 18, )

    mylist.insert(END, "Our Online Medical store management system will manage record of
all medicies , price details , avaliability of medicines , number of medicicnes sold ,
brand deatils , and other important details . It deals with the recording and  ")
    mylist.insert(END, "processsing data that medicicne store owners can easily manage
the operations .")
    mylist.insert(END, " ")
    mylist.insert(END, "Prescriptions are proper and supplied in precise amounts using
Medical store Management software. ")
    mylist.insert(END, " ")
    mylist.insert(END, "In this the various operations are used..!")
    mylist.insert(END, " ")
    mylist.insert(END, "Medicine Add :- In this we can add the refernce number and the
medicine name and as well as we can add , update or delete the datas in it .")
    mylist.insert(END, " ")
    mylist.insert(END, "Medicine Info :- In this we can check the type of
medicine,expiry date and much more and can add , update or delete the data of
medicine.")
    mylist.insert(END, "  ")
    mylist.insert(END, "Medicine Database :- In this we have the database of all the
medicines present in your store.")
    mylist.insert(END, " ")
    mylist.insert(END, "Daily Health Tip :- In this it will provide every time a new
health tip for motivating us to stay healthy.")
    mylist.insert(END, " ")
    mylist.insert(END, "Log Out :- In this simply clicking on it will logout you from
the tkinter window .")
    mylist.insert(END, " ")
    mylist.insert(END, "Our applications provides the following things :")
    mylist.insert(END, "  ")
    mylist.insert(END, "    1. To provide a login facility to the customers.")
    mylist.insert(END, "    2. To provide a list of all accessible medicines to the
customers.")
    mylist.insert(END, "    3. To update all the list of expired medicines pharmacist.")
```

```
    mylist.insert(END, " ")
    mylist.insert(END, " ")
    mylist.insert(END, "The key characteristics of Online Medical store Management
System Project can be outlined as:")
    mylist.insert(END, " ")
    mylist.insert(END, "    1. It implements both hardware and software during the
management of memory.")
    mylist.insert(END, "    2. The project limits the programs from immediate closure
when the RAM is full.")
    mylist.insert(END, "    3. It makes the CPU (Central Processing Unit) store excess
processing data on a hard disk.")
    mylist.insert(END, "    4. It is an economic step in computing architecture.")
    mylist.insert(END, "    5. Its mapping memory addresses which have been used by
different programs.")
    mylist.insert(END, " ")
    mylist.insert(END, "Thanks!")
    mylist.insert(END, "Regards:")
    mylist.insert(END, "Medical store management developer team")

    mylist.pack( fill = BOTH )
    scrollbar.config( command = mylist.yview )
#Exit button
def logout():
    root.destroy()

#main frame
Frames1 = Frame(Frames, bd = 15, relief = RIDGE)
Frames1.place(x=0, y = 100, width = 1450, height = 500)


img = ImageTk.PhotoImage(Image.open("fav.png"))

# Create a Label Widget to display the text or Image
label = Label(Frames1 , image = img)
label.pack()

btnadd = Button(Frames1,text="LOGIN",  font = ("times new
roman",12,"bold"),fg="teal",command= login,
                    bg = "white" ,padx = 2, width = 20)
btnadd.place(y = 250, x = 1000)

root.mainloop()
```

```
connectiondb.close()
```

Login page:

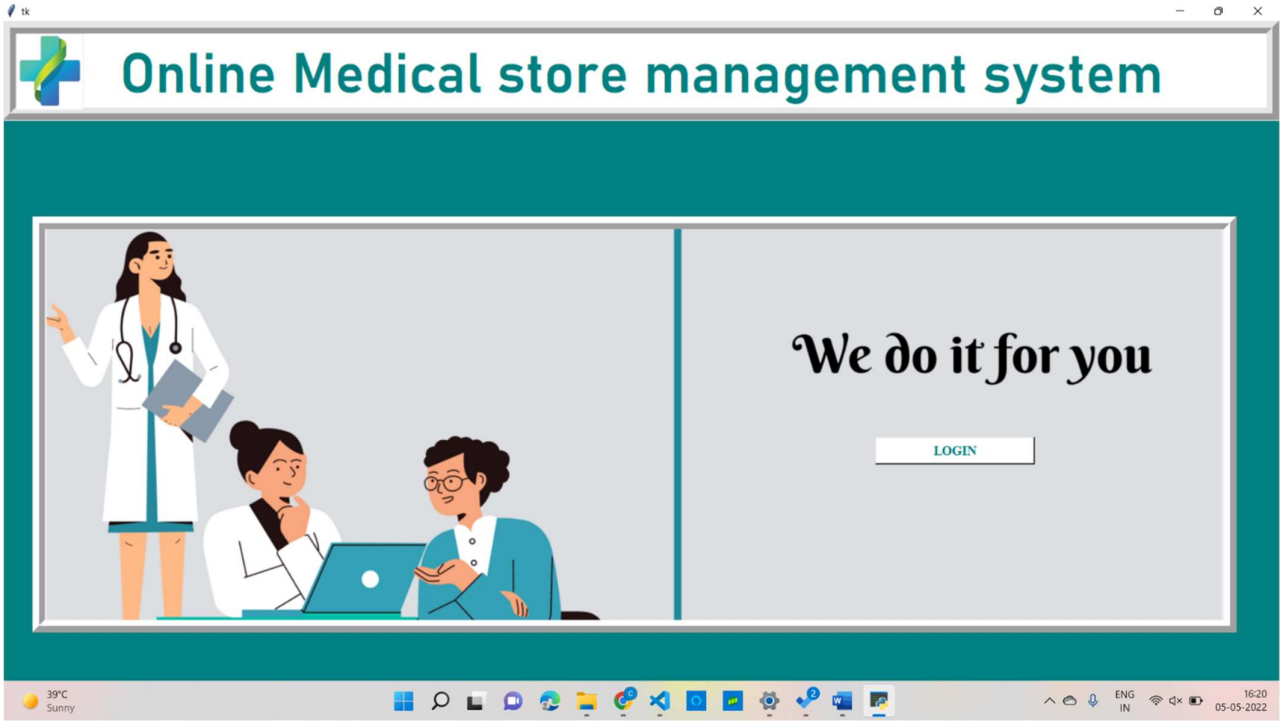


Fig. R.1

This is the landing page of the application that we have created and here is the login button to get you inside the application.

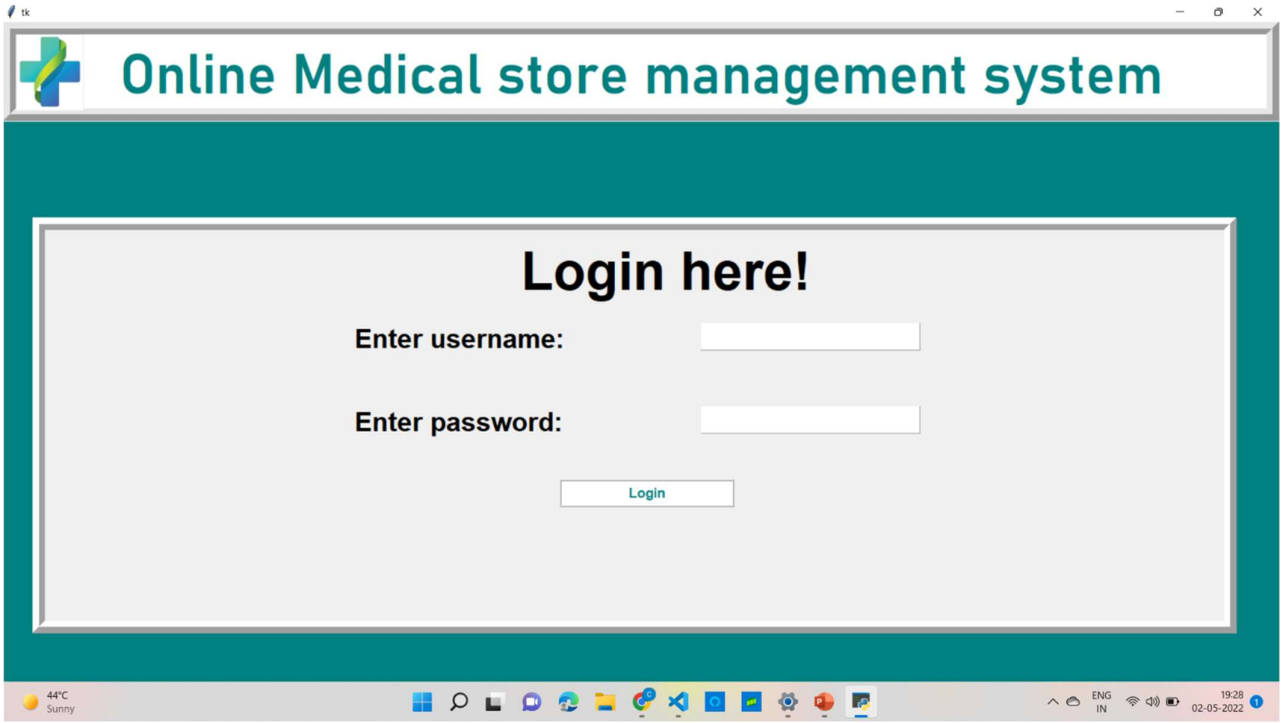
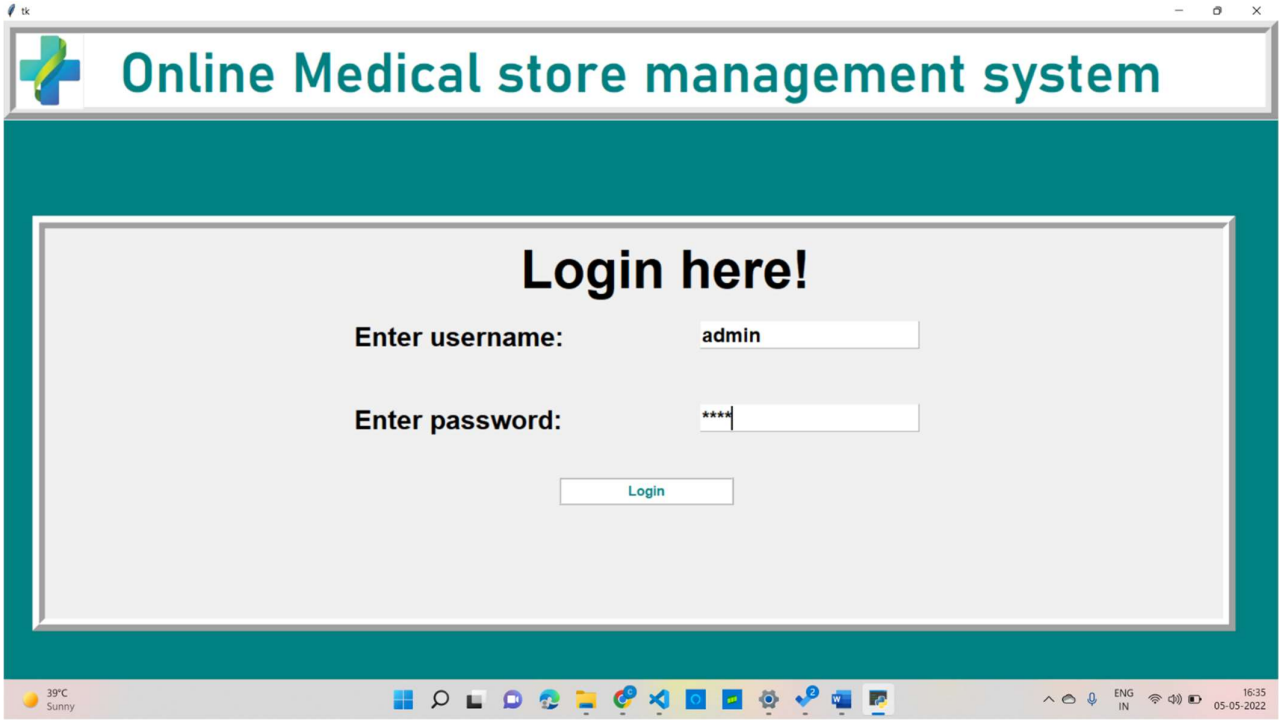


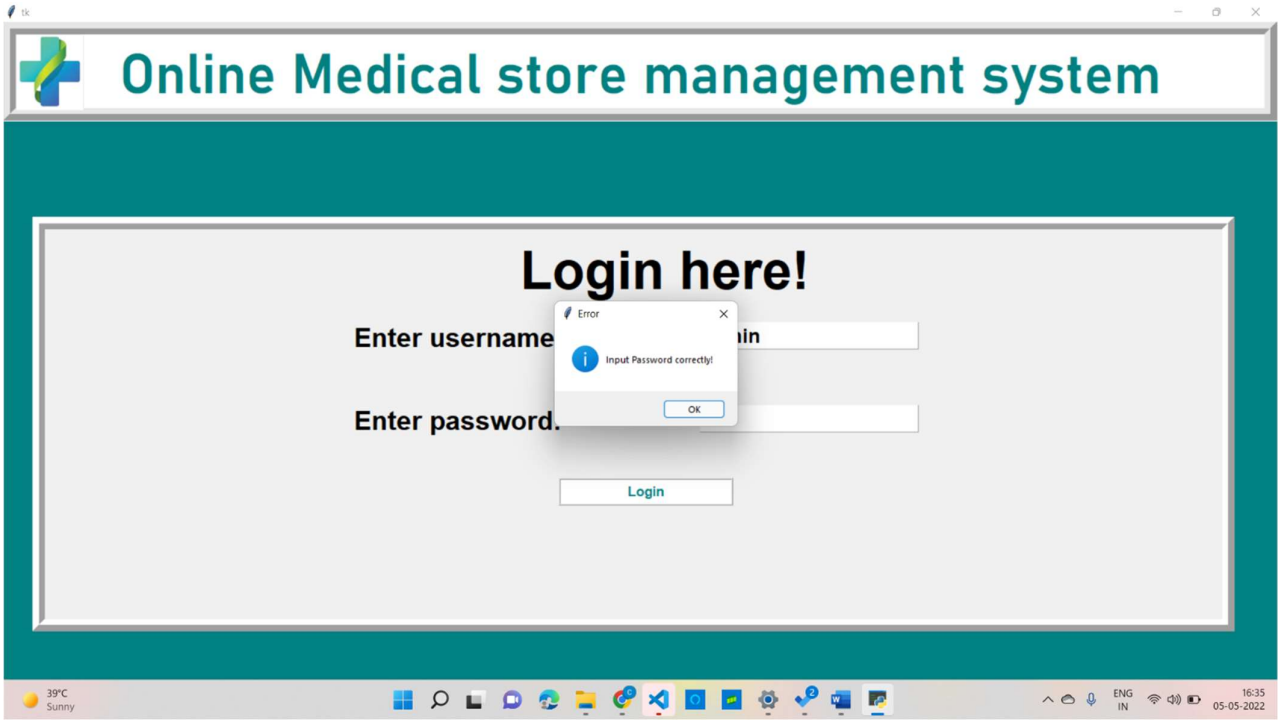
Fig. R.2

This is login page of out system. You have to enter the login in the application before performing any task

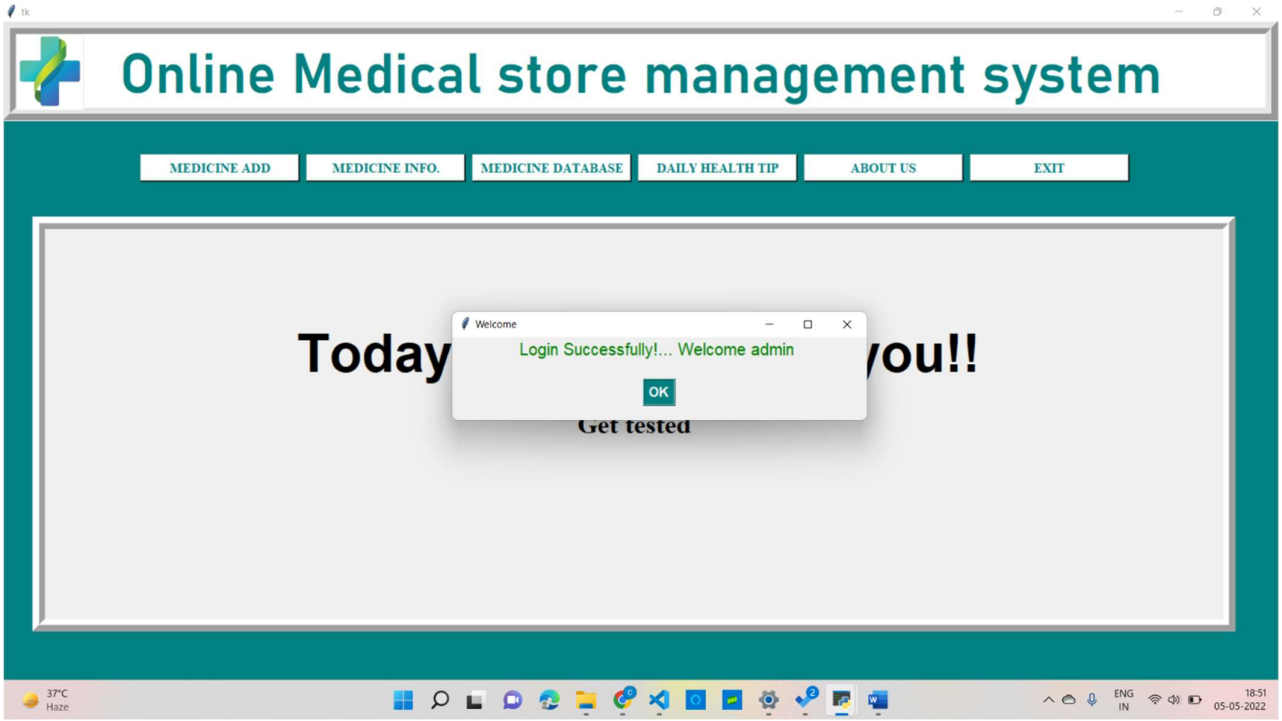
in it. (NOTE: Login credentials are always unique and confidential)



Here, you can enter your login details. The system is designed in such a way that your password will remain confidential if you enter it in front of anyone as * will be displayed in place of entered alphabets or numbers.

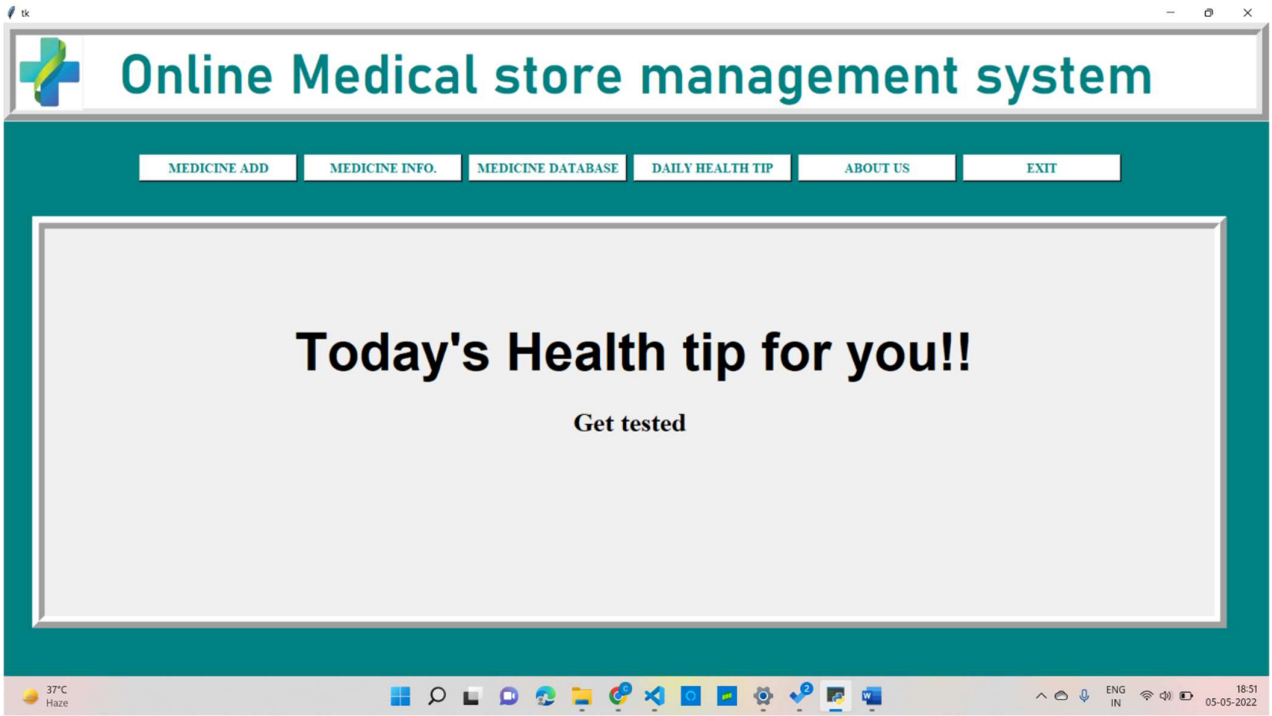


This is the view to show the pop-up message that will appear in case you entered the inappropriate username or the password. So, enter the password and username correctly.



This is the first screen that will appear after you get logged in the system. Here, you can see the pop-up screen that will appear where it is written to welcome you:

Login Successfully!... Welcome <username you have entered>



Click OK on the above screen and now your screen will look like this. This is the page that will show the health tip once you get logged in the system. This is designed in such a way that it will always show different message as the health tip.

Above there are some buttons that you can see, these are the navigation buttons that have different functionalities. There is short description of all those medicines.
The description is as follows:

1. **Medicine Add:**

   This button will take you to the page that is required to give a unique reference number to the medicines in the shop. In medicine shops reference number is used to identify different medicines uniquely.

2. **Medicine info:**

   This button will take you to the page that is required to add details of the medicine like, from where the medicines are bought, quantity of medicine bought, price, issue date, expiry date and other many more details that one considers while buying any medicine.

3. **Medicine database:**

   This is the tabular view of the database that is saved using the medicine info. Column and This is used to search the medicines in the medicines. You can search medicines based on different options like reference number, medicine name or the lot number.

4. **Daily health tip:**

   This is the column where we have added different medical tips that will help you to boost your daily routine and good health. Here are some statements that we have added:

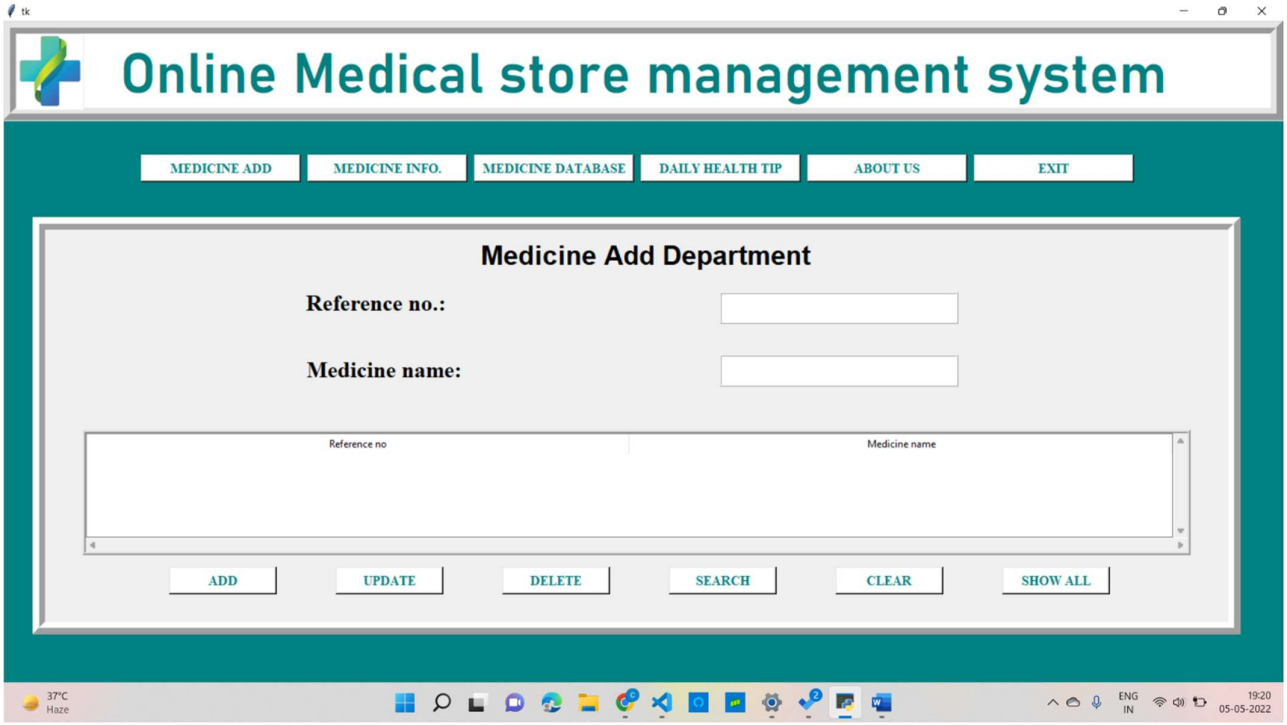
   - Get tested.
   - Check blood pressure regularly.
   - Take antibiotics only as prescribed.
   - And many more…

5. **About:**

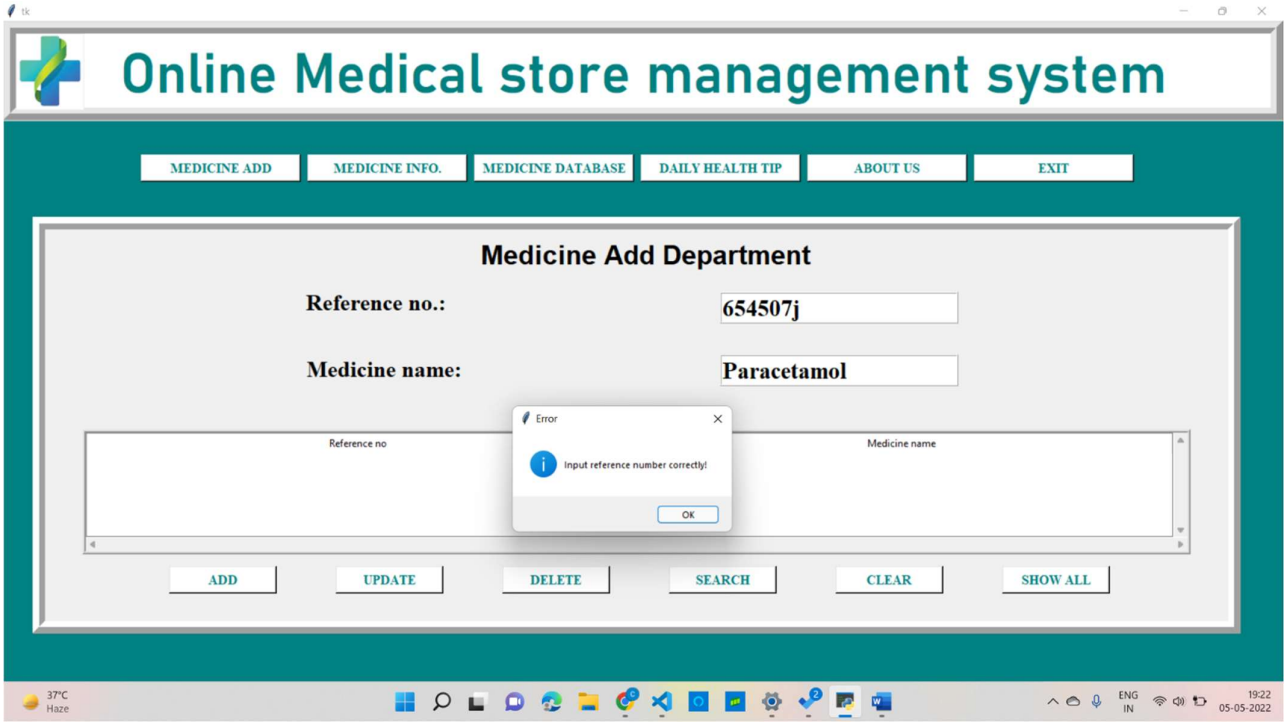
   This is added where you can get the details about the basics of the medical store management system. Here is the details of the functions that our medical store management system can perform. In simple words, it contains a guide to our medical store management system.

6. **Exit:**

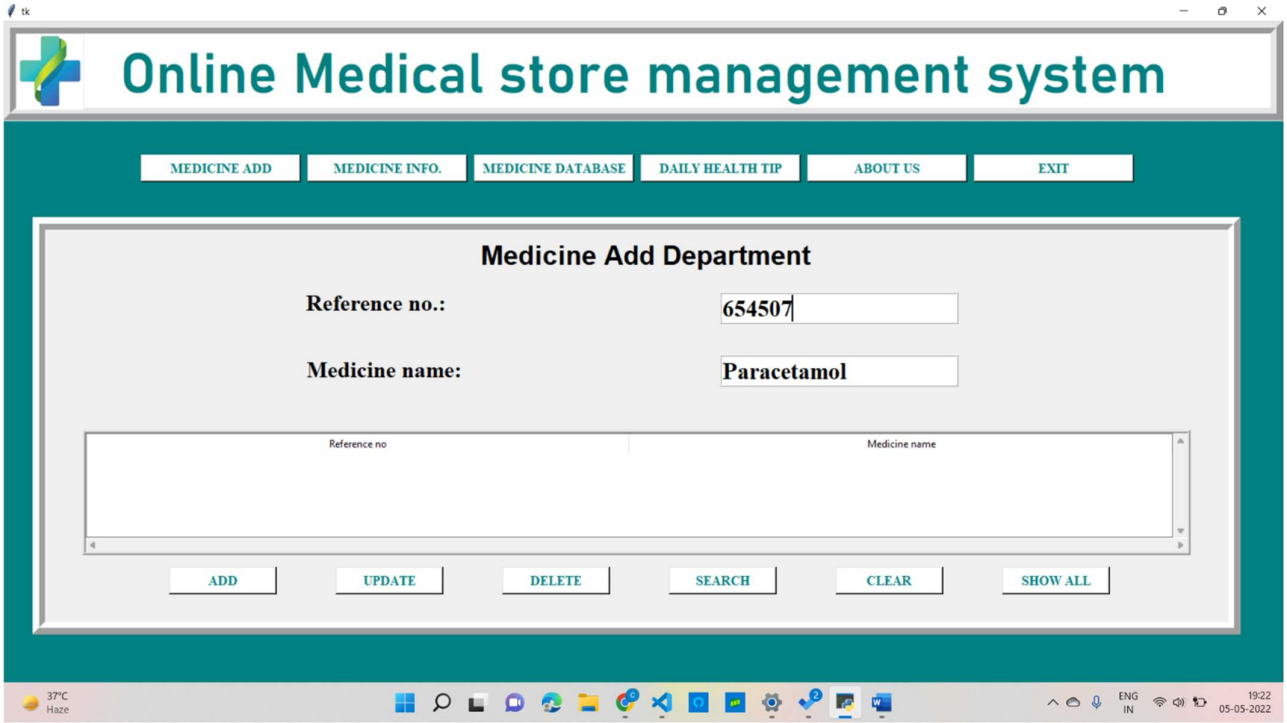
   This button came in use when you are done with the use of the application and now you want to close the functionality of the system. On pressing this button, the root or the destroyed and next time, you have to run the application and login again also.



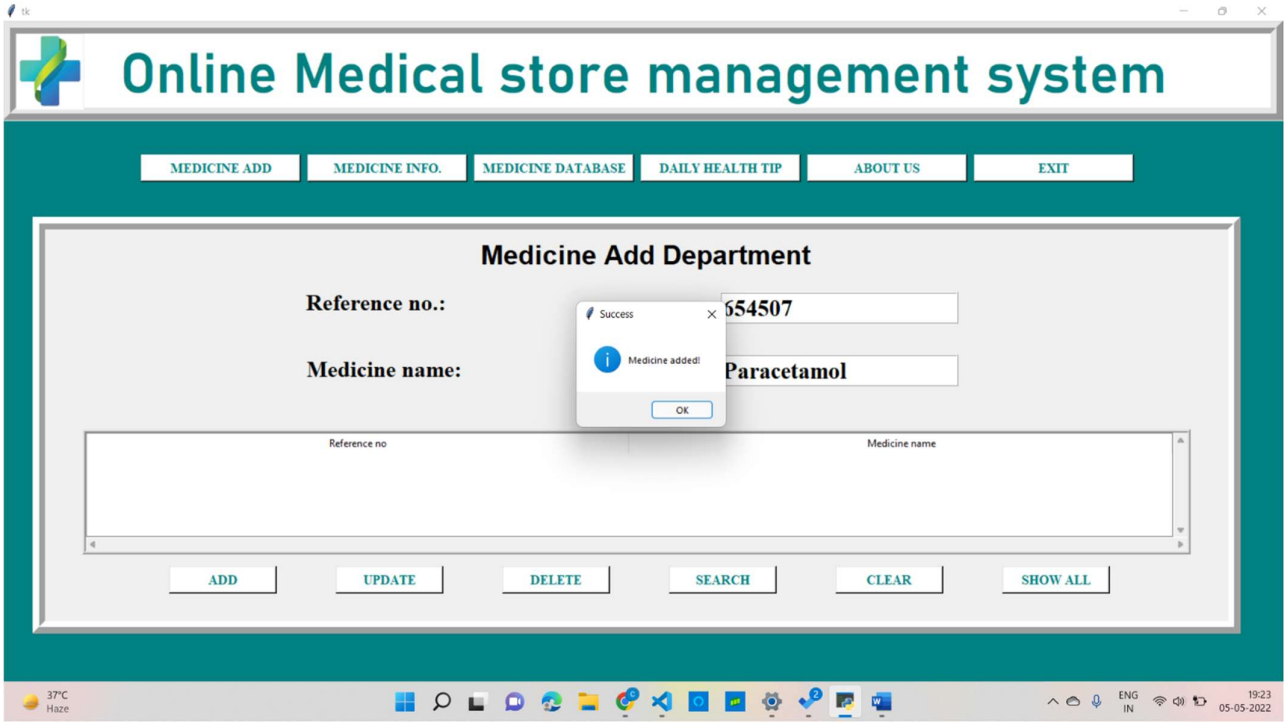
This is the screen that will get open on pressing the medicine add button and this is used to give a reference number to the medicine.



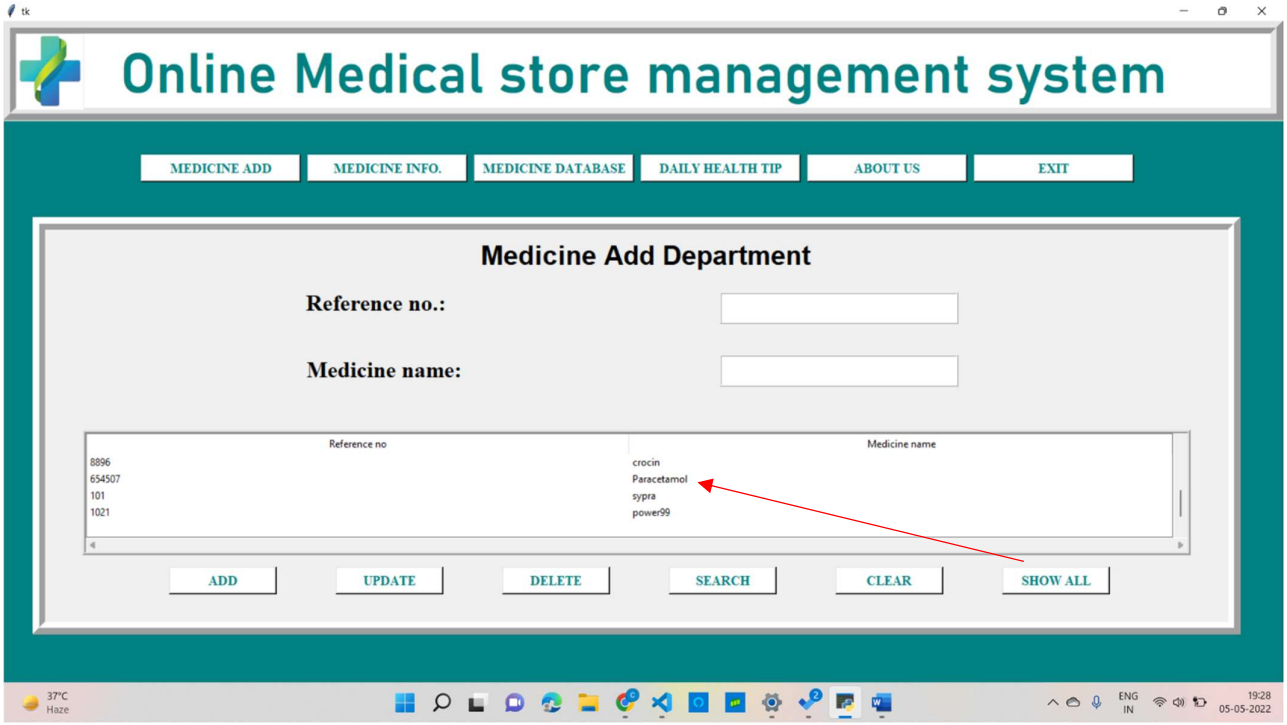
In this, you have to enter the reference number and medicine name only. Reference number should be in the number format only. And if you enter the data in wrong format like above, the error message will pop-up then.



So it is requested to enter the data in the correct format as shown in above picture. After this, click on add button to add medicine. And it will appear in the database then.

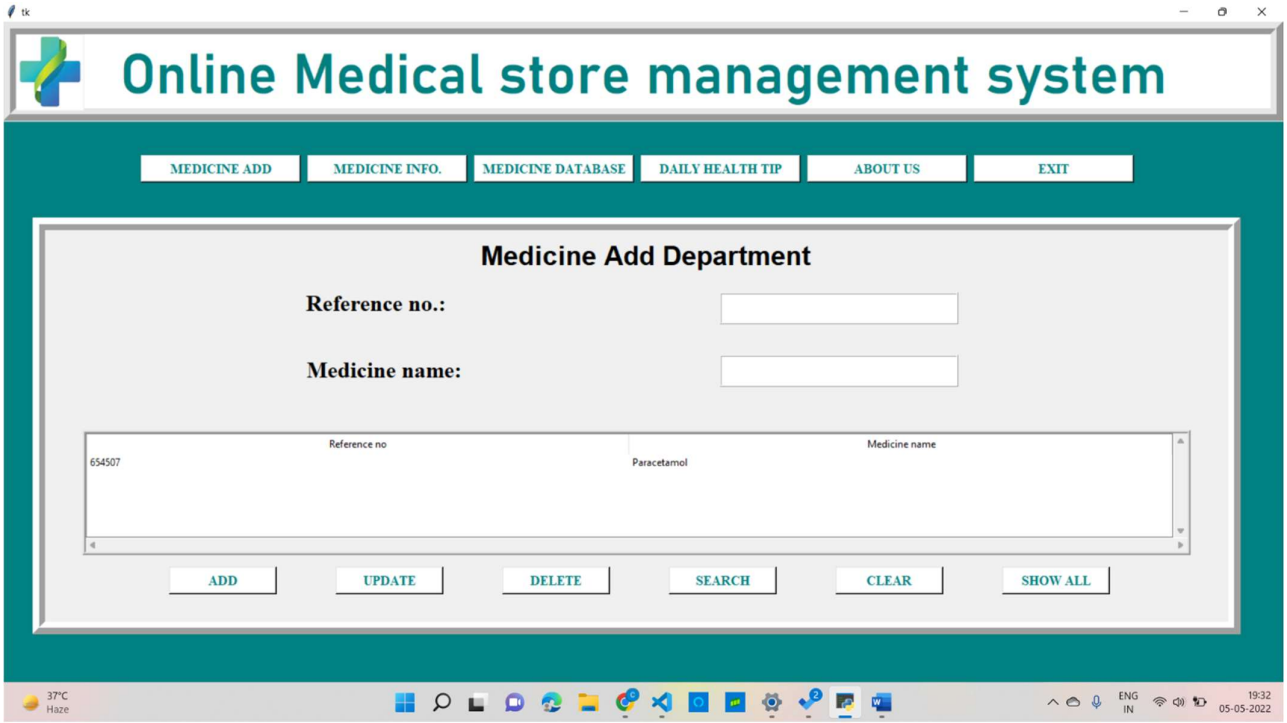


Click on the show all button to show the database and find your medicine in it.

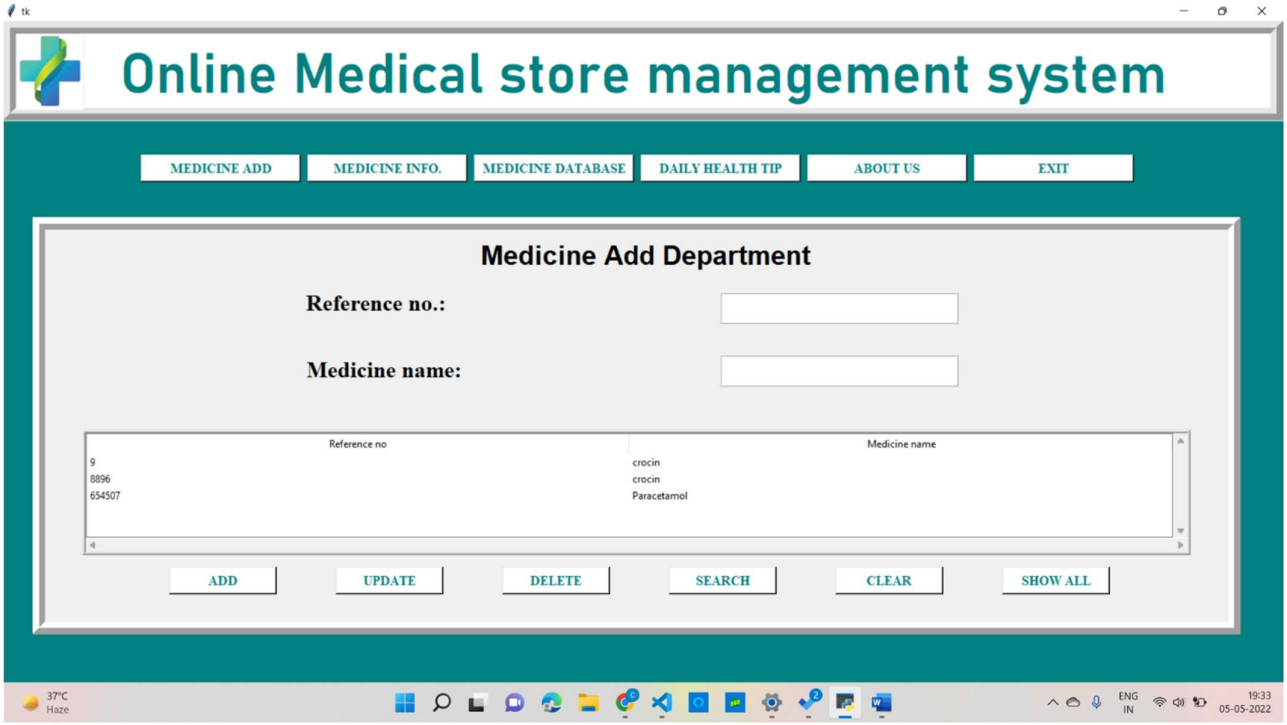


You can also search the medicine using the search button. In search button, you need to add reference number and medicine name and if they are not matching simultaneously, then we will get the details of the both medicines based on reference number and medicine name. Both the cases are shown below:

→ On putting 654507 and Paracetamol respectively then:

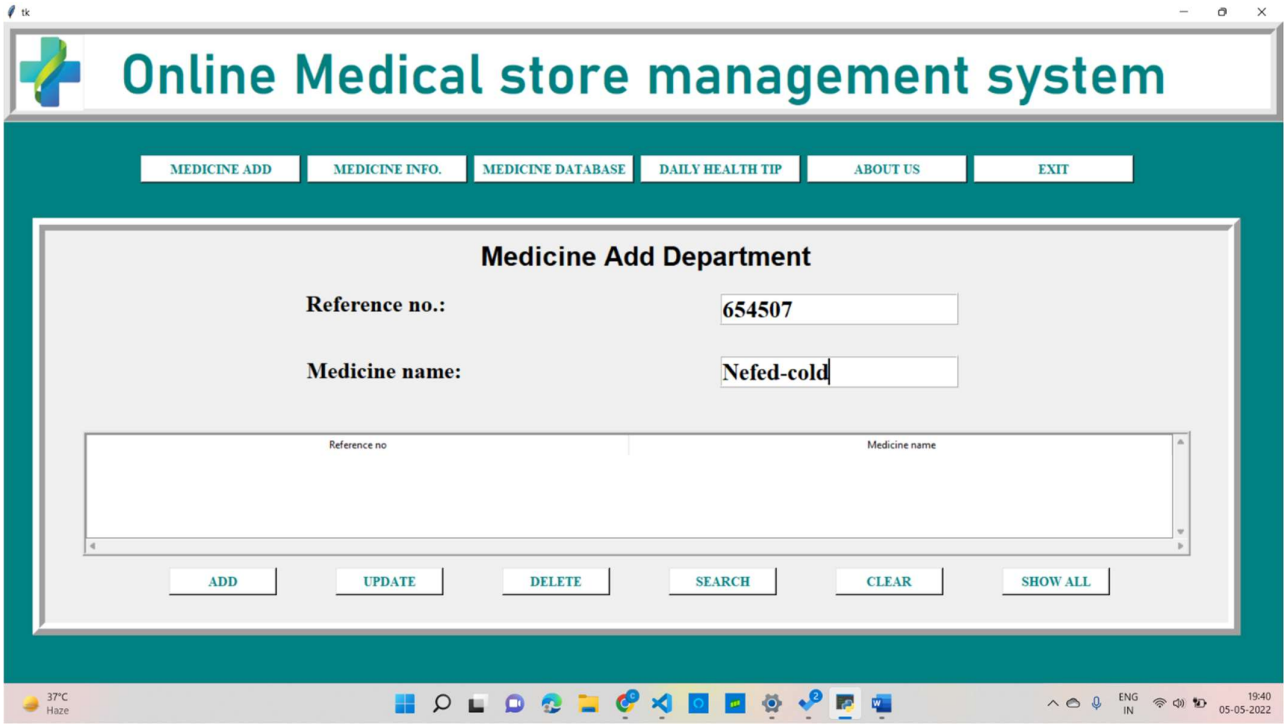


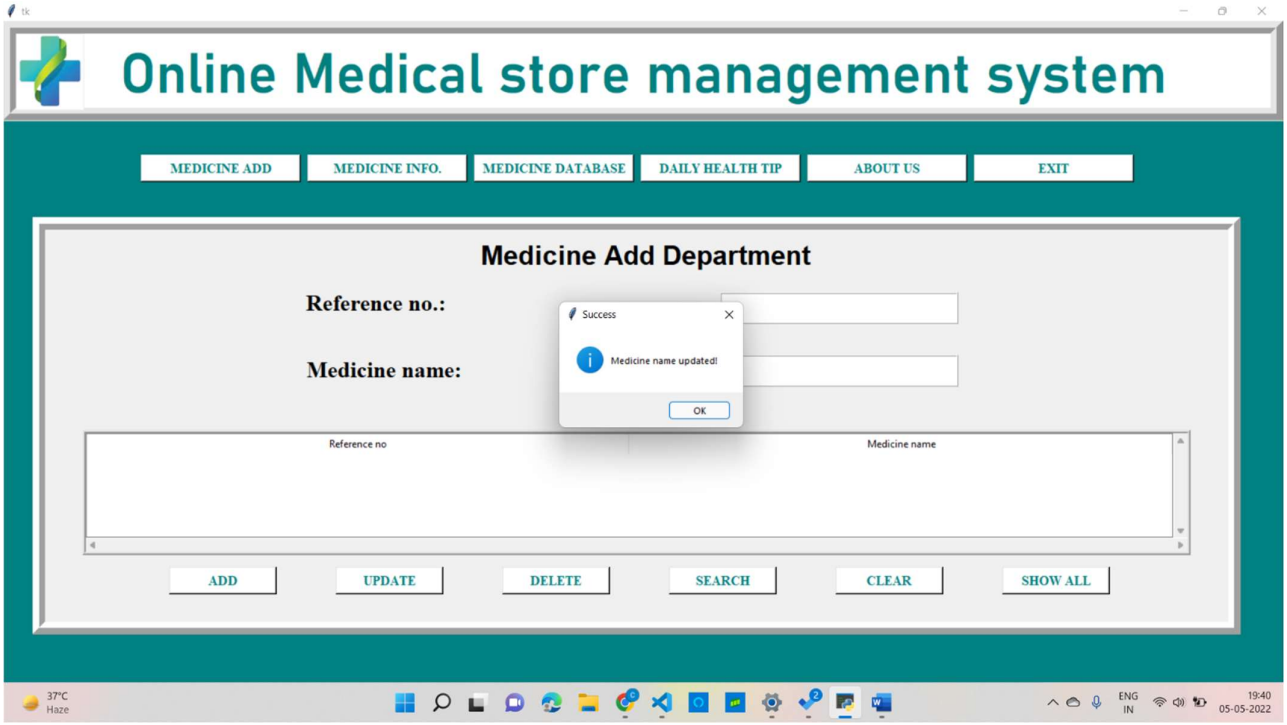
→ On putting 654507 and crocin then:



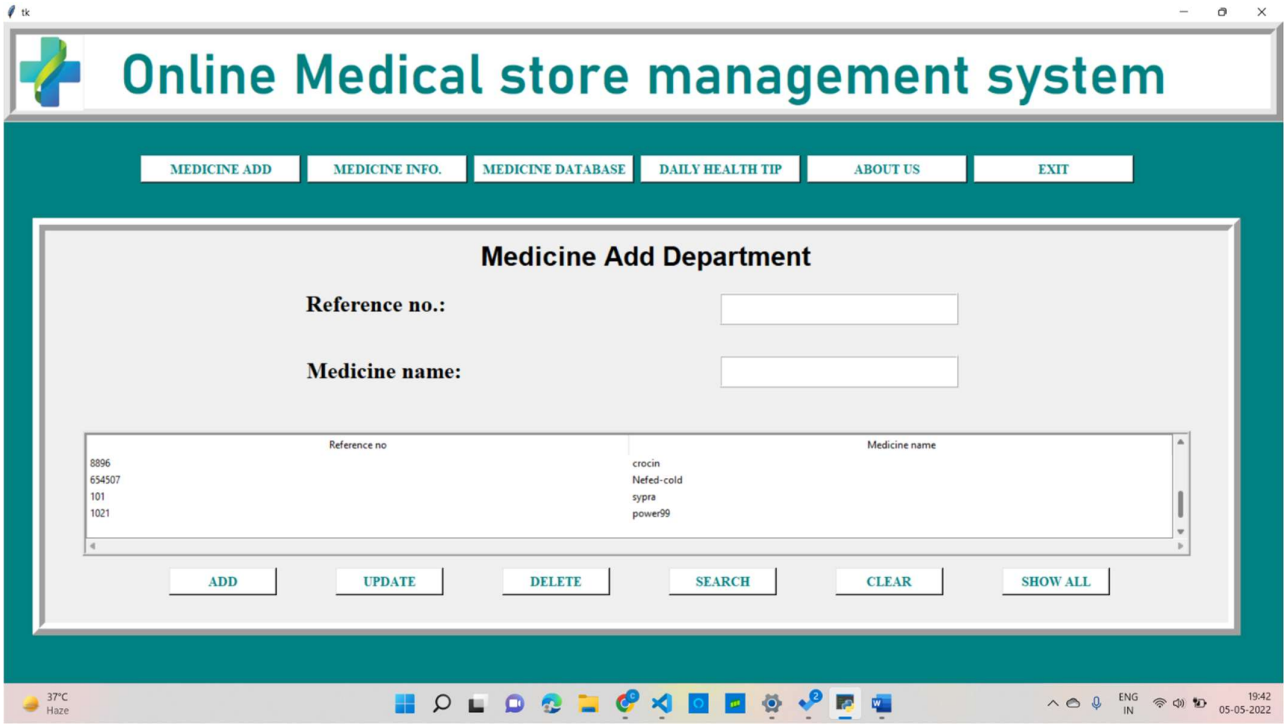
We can also update the medicine details like we have Paracetamol at 654507 so we need to replace paracetamol with Nefed-cold… so the procedure is as follows:

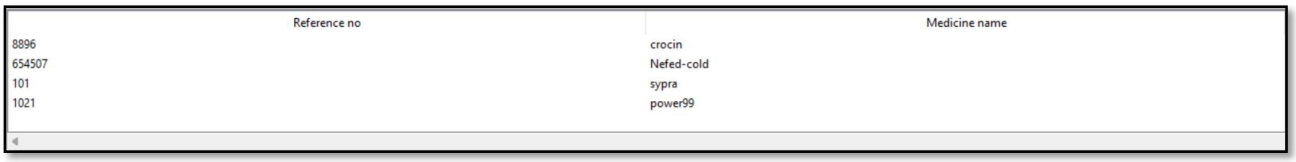
Write the reference number and the new medicine name in the place so it can be updated to get the result.





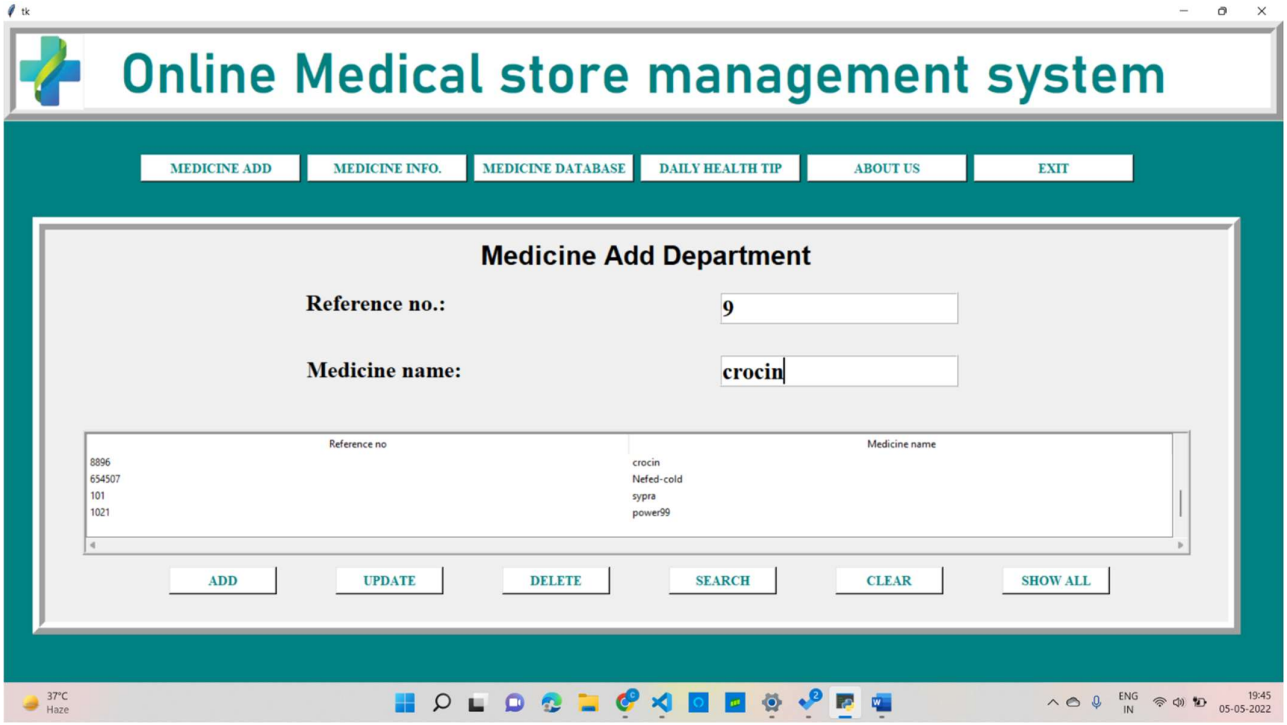
Now you can also search or may check the database about the medicine:



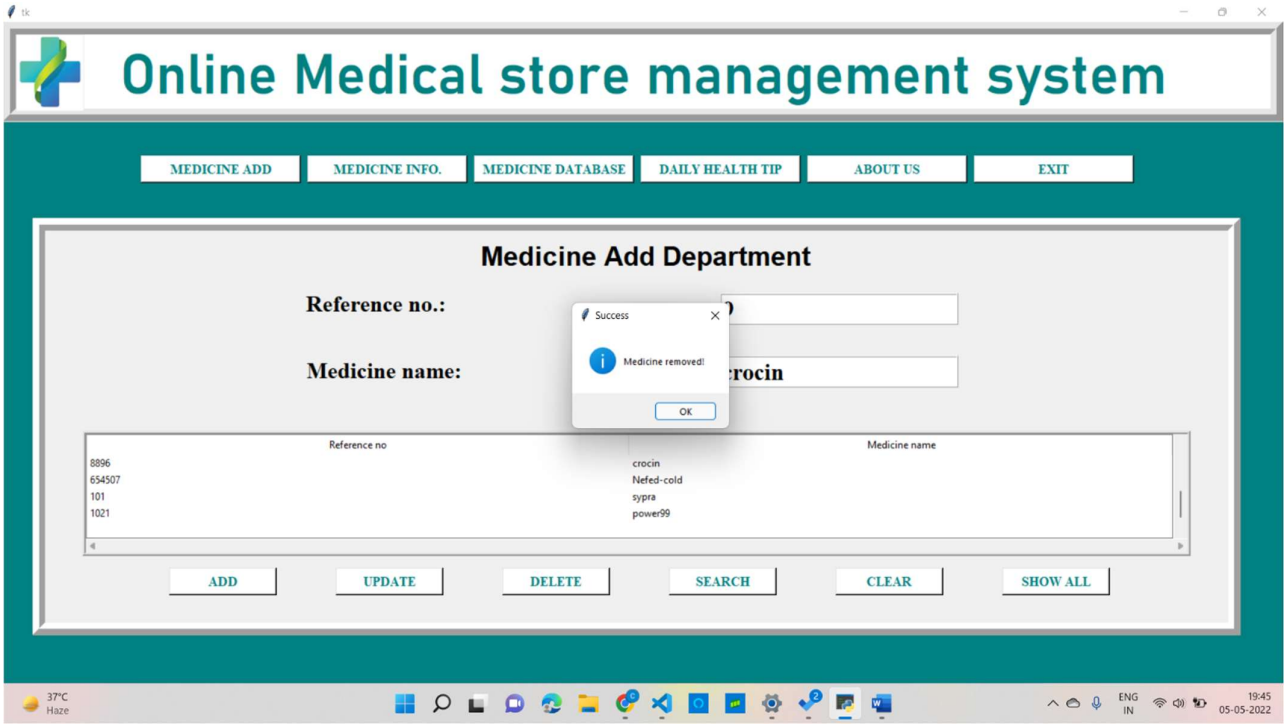


In the above example of search, you have observed that each medicine has unique reference number but the crocin is present 2 times in the database. So, we need to delete that:

Enter the details of the medicines as follows and delete it:



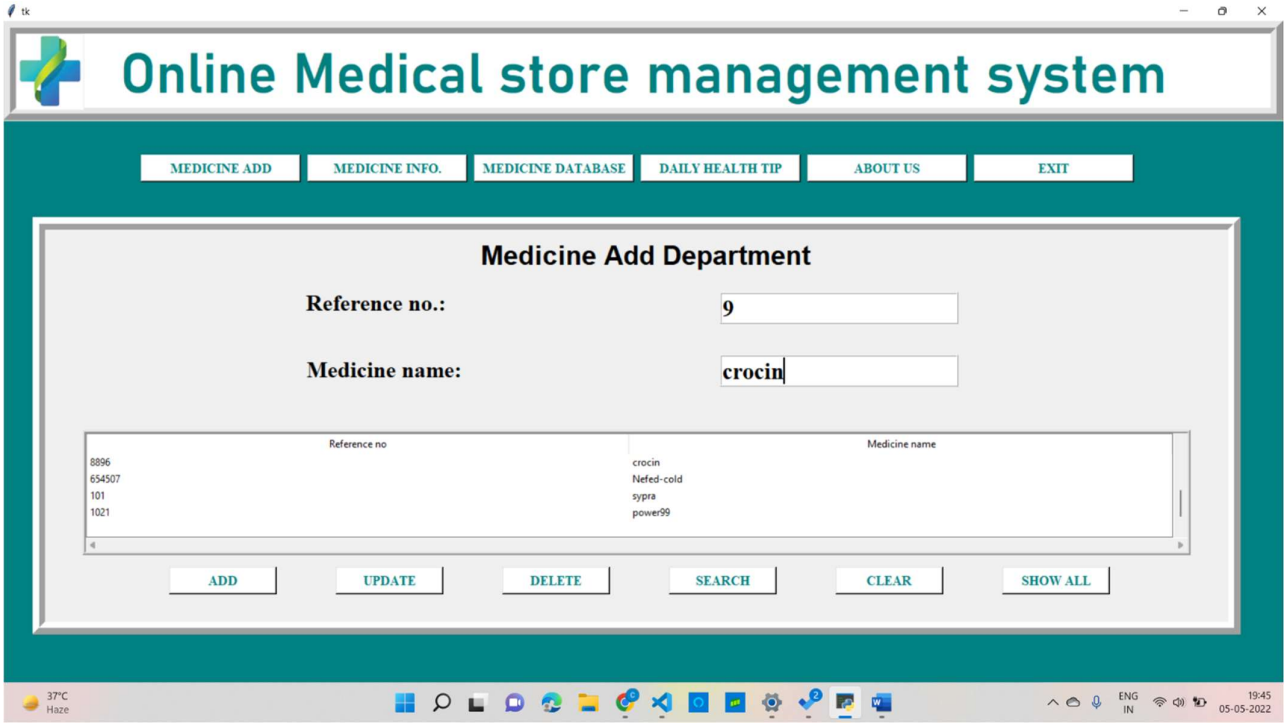
Hence, the medicine is removed from the system.



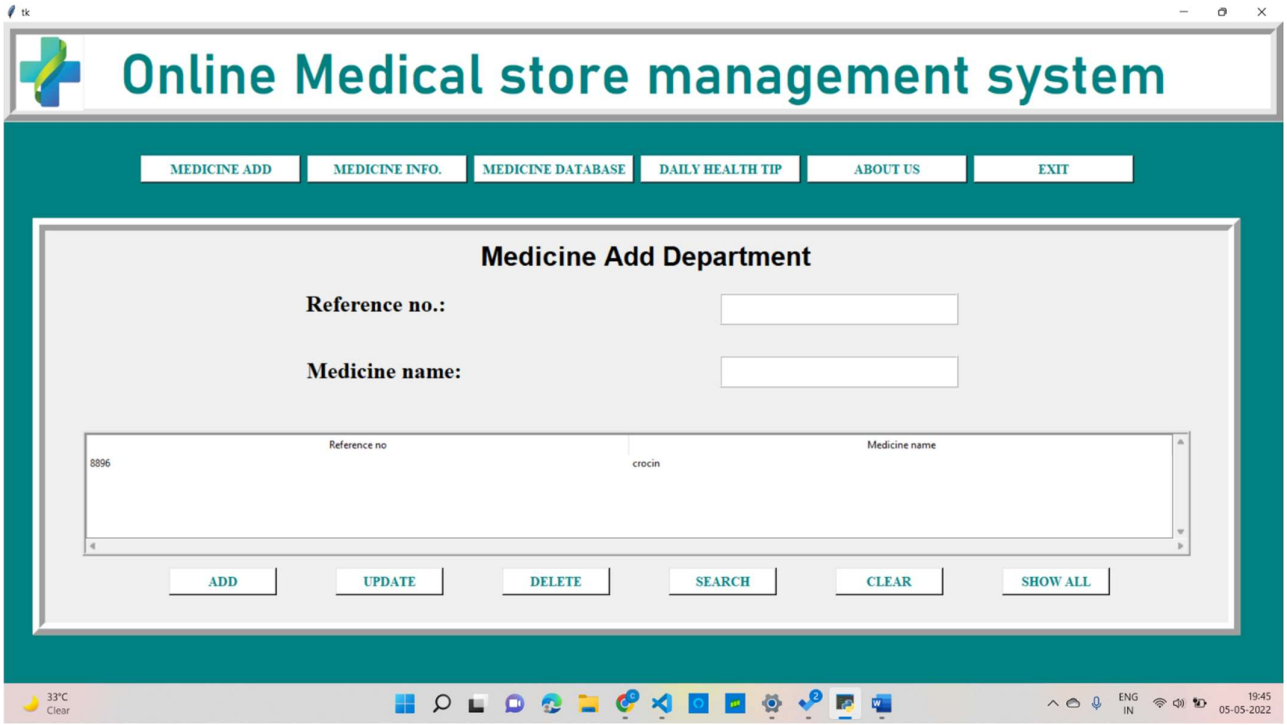
Now, search the medicine again in the database, here is only one reference number for crocin:

→ So, now again enter the following details:

Reference no.: 9
Medicine name: Crocin



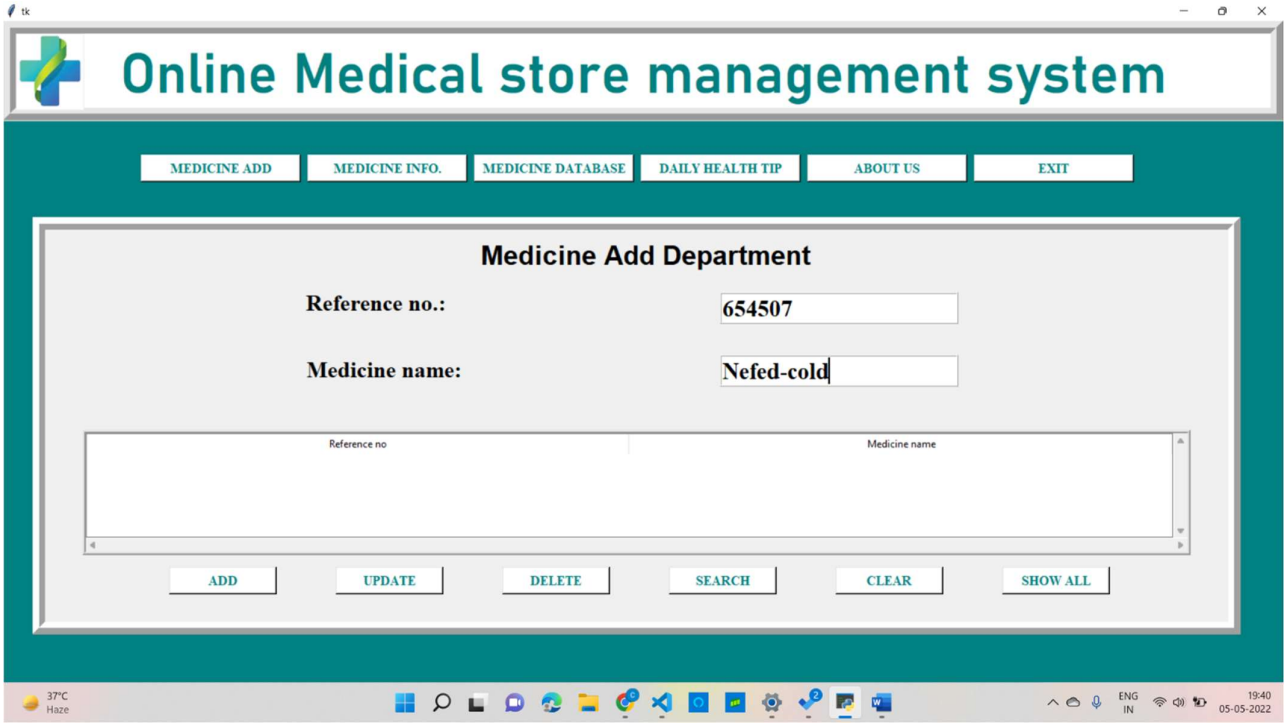
Entered the above mentioned details as follows and now its time to search.



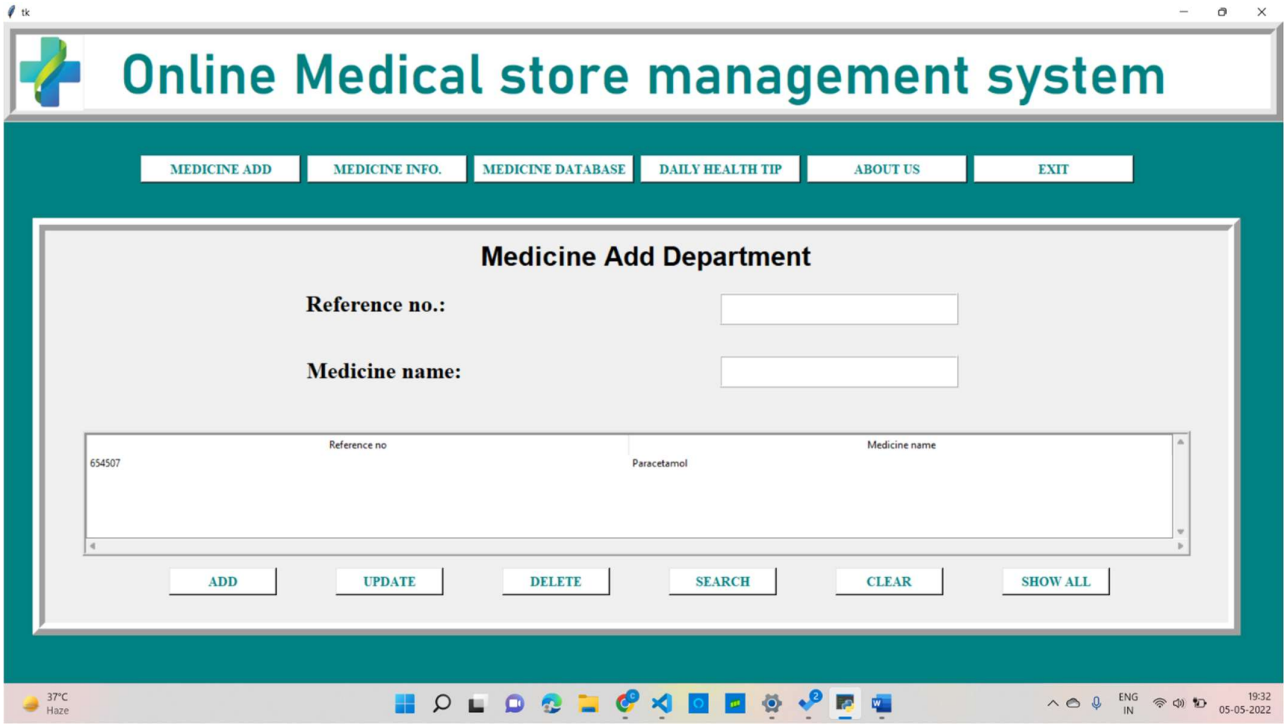
There is no medicine at reference number 9 and now, the crocin is also present at reference number 8896 and here are the results.

At last clear button is used to clear the reference number and medicine number as written in the blank. So that it can be entered again then.

It is as follows:



On pressing the clear button, entered value will get disappear and blanks will be again there.



This is the complete details about the medicine add function of the application and we have now aware of about the use of this function and how this function act.

On a short note, its functionalities are:

- ➔ Add a medicine and reference number.
- ➔ Update a medicine name with respect to reference name.
- ➔ Delete the medicine with its reference name.
- ➔ Search a medicine in the database.

- ➔ Clear the text already entered in the blanks.
- ➔ To show the complete database.

Now its time to look out the functionality of the second button that what the 2nd button can be used for.

The second button in the navigation column is Medicine info. And that is used to add the following details about the medicine:

- Reference number
- Type of medicine
- Company name
- Medicine name
- Lot number
- Issue date
- Expiry date
- Side effects
- Precautions
- Dosage
- Price
- Quantity

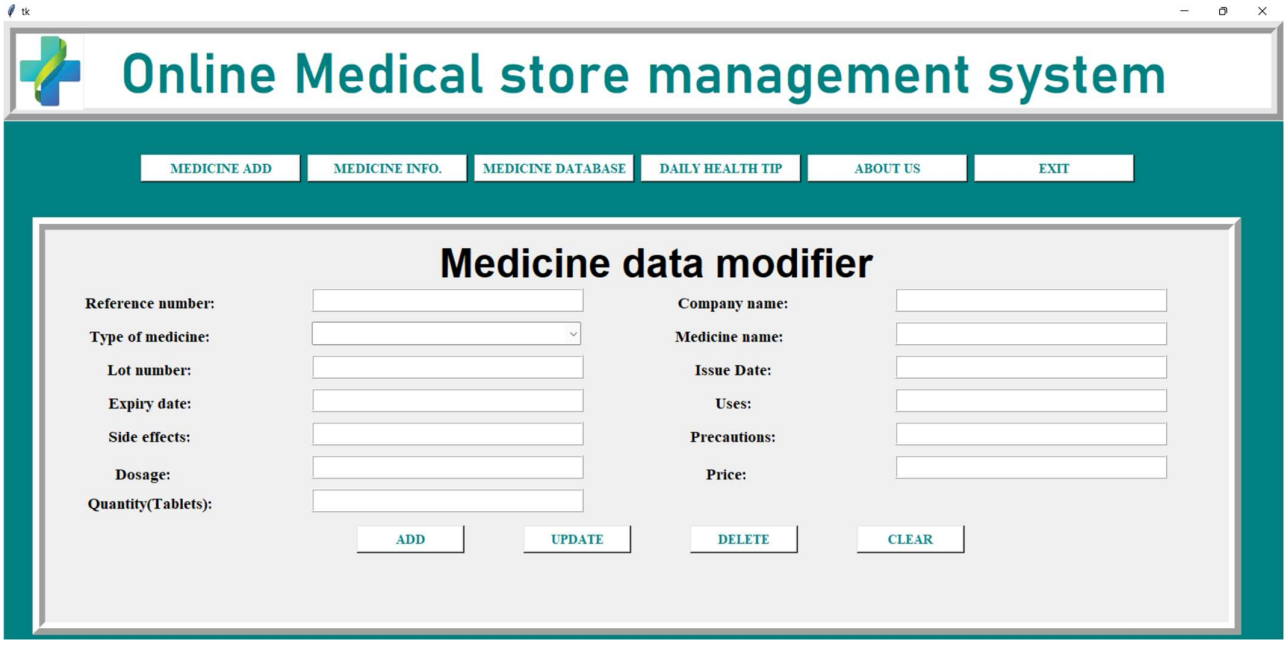
And the functions that we can perform on these operations are as follows:

- Add
- Update
- Delete
- Clear

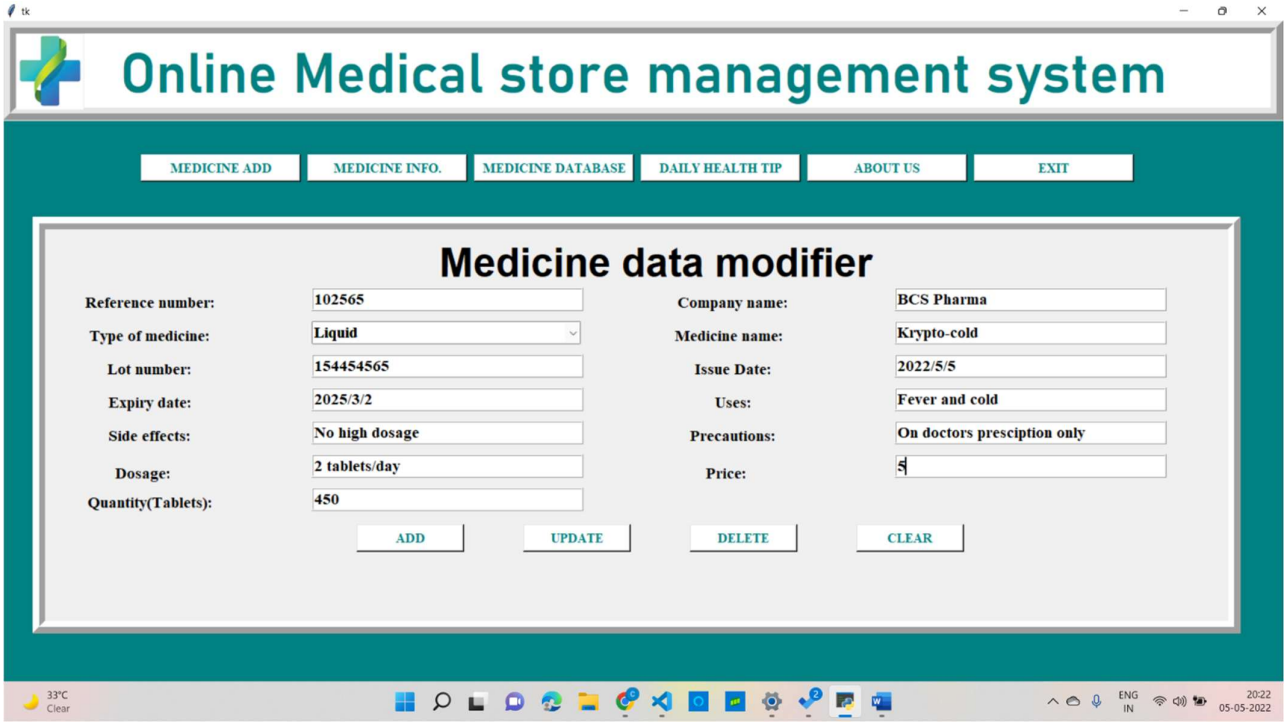
The task of this page is as follows:

- To review about the dealers of those medicines.
- To know about the issue and expiry date of those medicines.
- To keep an eye on the side effects and precautions of the medicines.
- To check the price of the given medicine.
- To check the quantity of the medicine available in the medicine shop.
- To know about the prescribed general dose of the medicine.

Here the page actually looks like this:



Enter the details are as follows:



There are few constraints that we have applied to each of the blanks.

- **Reference number**
  - It can't be empty.
  - It should contain text in integer format only.
  - No other format containing alphabets or symbol will be accepted.
  - Here, it is the foreign key but will not act as primary key.
- **Type of medicine**
  - It will contain only the specific format as mentioned in the scroll bar. The items in the scroll bar are as follows:

- Tablet
- Liquid
- Tropical medicine
- Capsules
- Drops
- Inhale
- Injections

- **Company name**

  This contains the name of the company only.

- **Medicine name**

  This will contain the name of the medicine only.

  NOTE: This key will act as the foreign key so you should not add the different medicine as corresponding to different reference number.

- **Lot number**

  This will contain the lot number of the medicine. Note the point that this will contain the text in the integer format only. This key will act as the primary key here.

- **Issue date:**

  This will contain the date on which the medicine has been issued. Note the point that this will contain the date only in the format: YY – MM – DD

- **Expiry date:**

  This will contain the date on which the medicine has been issued. Note the point that this will contain the date only in the format: YY – MM – DD

- **Side effects:**

  This will contain the side effects of the medicine in the text format.

- **Precautions**

  This will contain the precautions of the medicine in the text format.

- **Dosage**

  This will contain the dosage of the medicine in the text format.

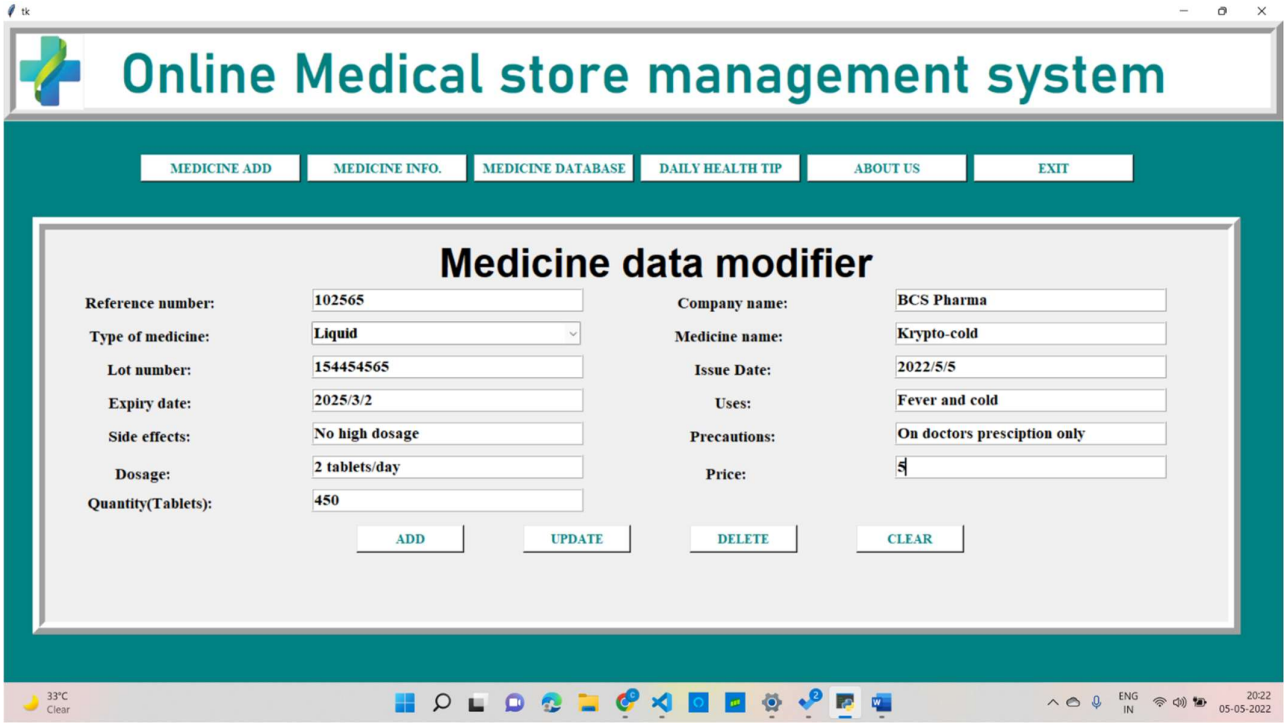
- **Price**

  This will contain the price of the medicine per tablet in the text format. Note the point

- **Quantity**

  This will contain the quantity of medicine available per tablet/capsules/bottles/etc.

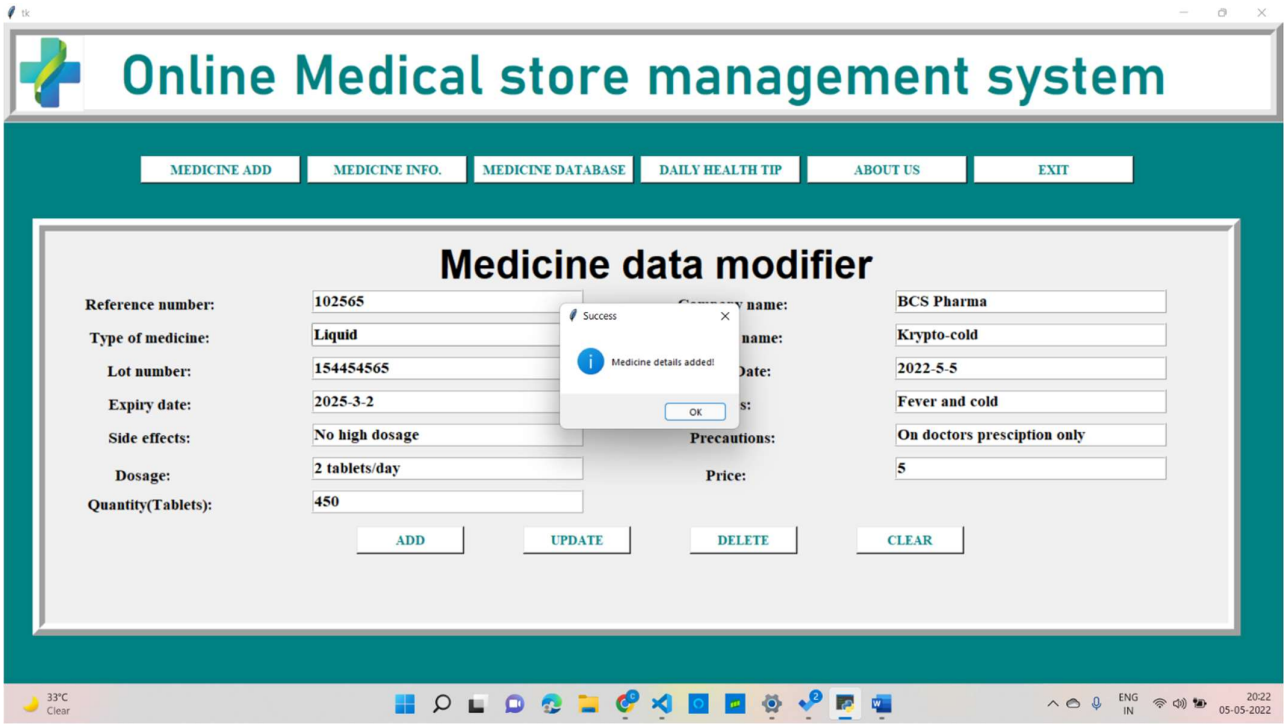
This is the format of writing the details of the blanks that we have to fill while adding the details about the medicine information. Here is the correct format of writing the medicine details in the column:



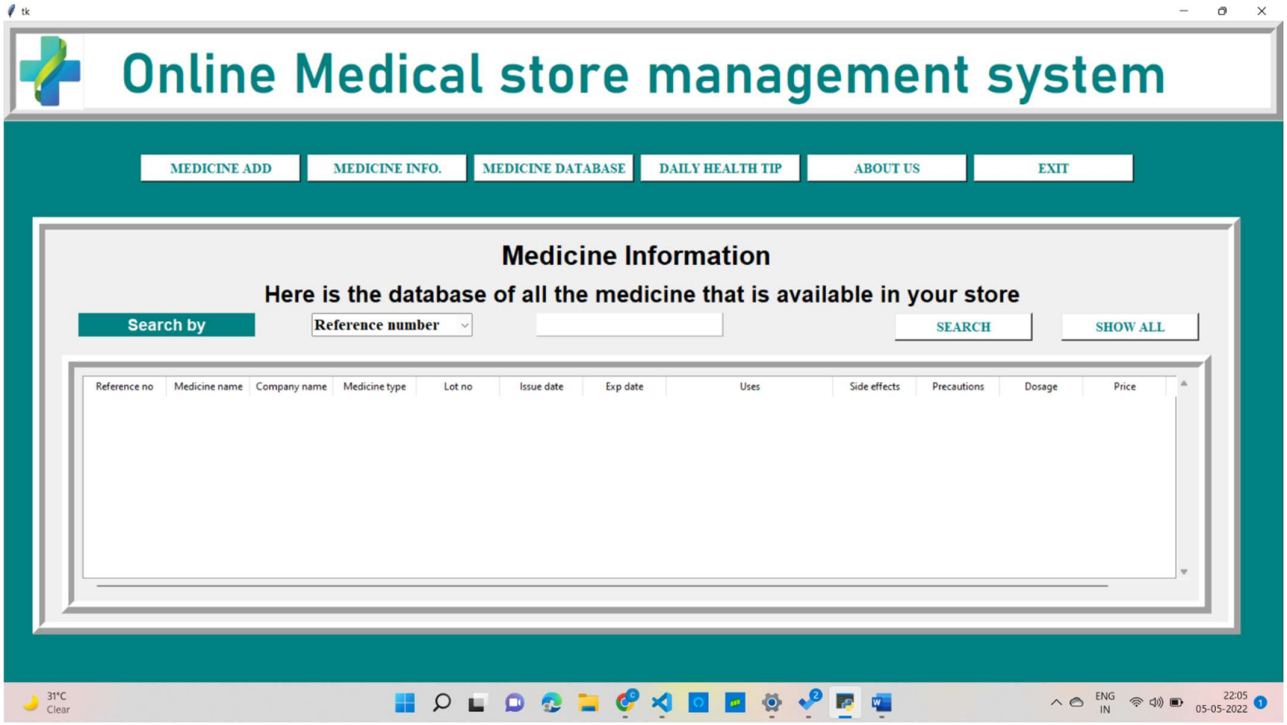
And after adding the medicine, it will be saved in our database.

Note the point:

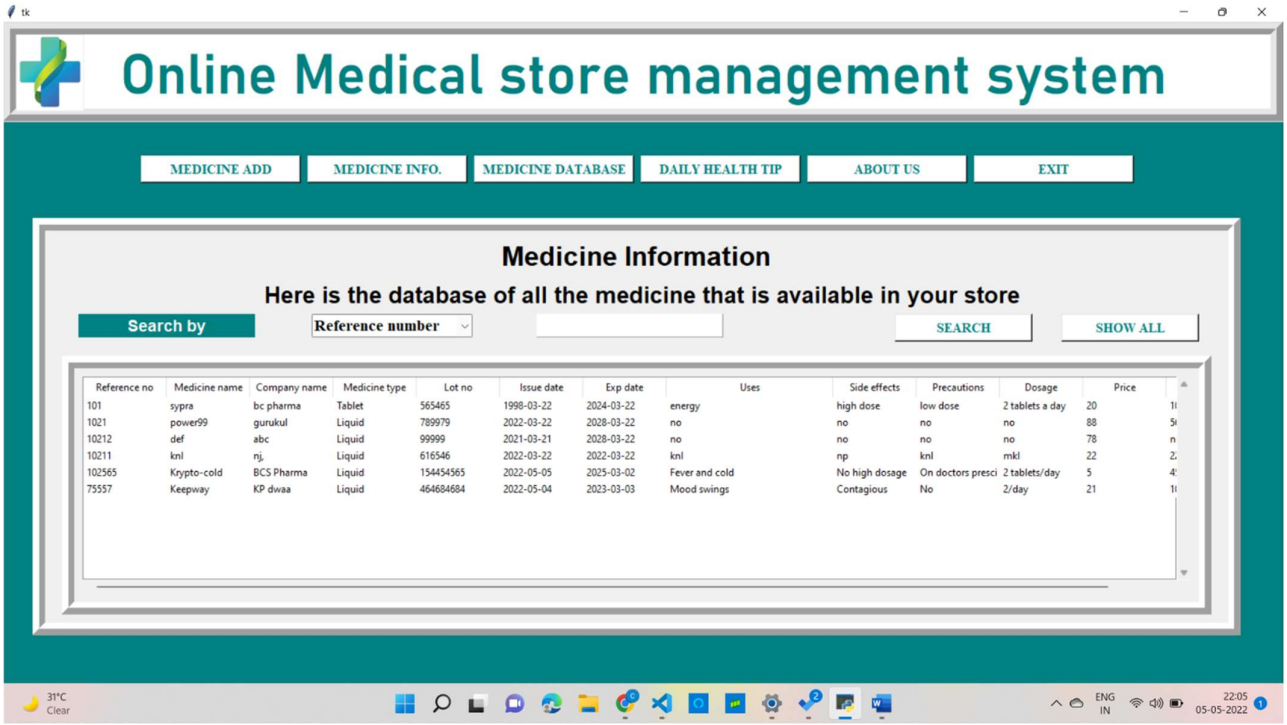
One should need to write the data in the specified format only.



Now, you can navigate to the medicine database button where you can go for show all button where you can check the medicine that is added in the database:



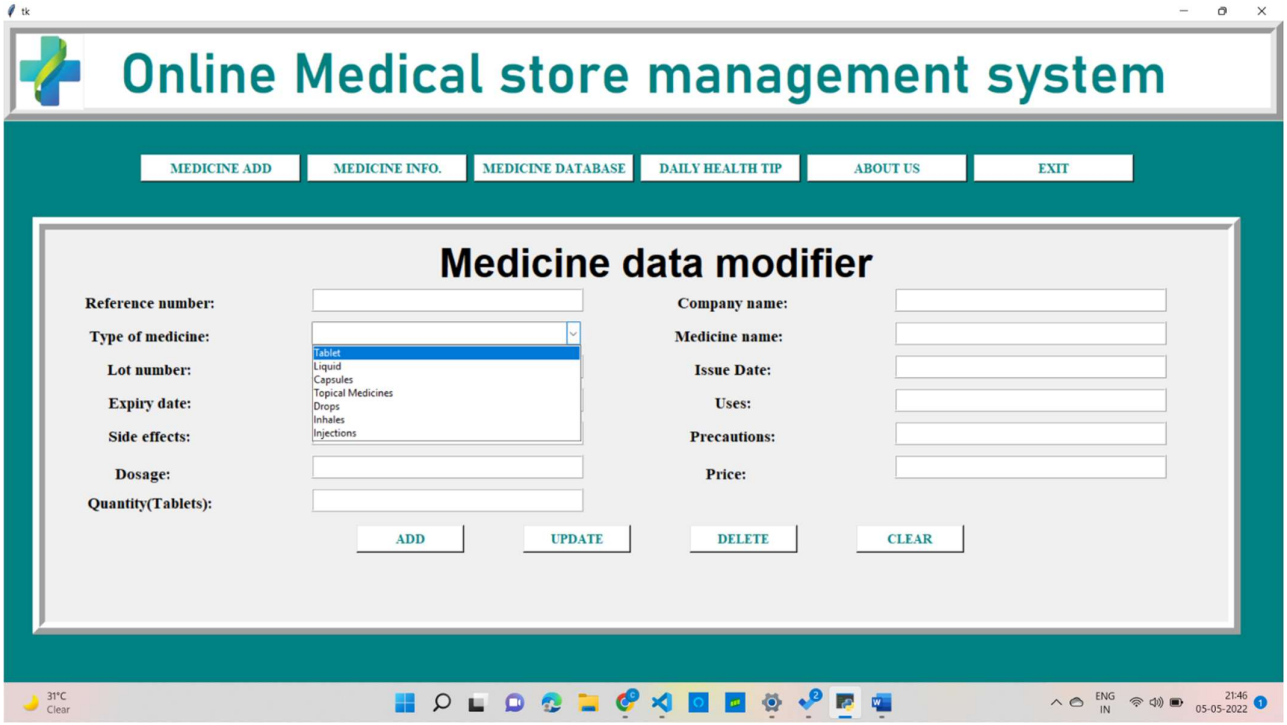
Click on show all button to display all the medicine details that are available in the database.



102565	Krypto-cold	BCS Pharma	Liquid	154454565	2022-05-05	2025-03-02	Fever and cold
75557	Keepway	KP dwaa	Liquid	464684684	2022-05-04	2023-03-03	Mood swings

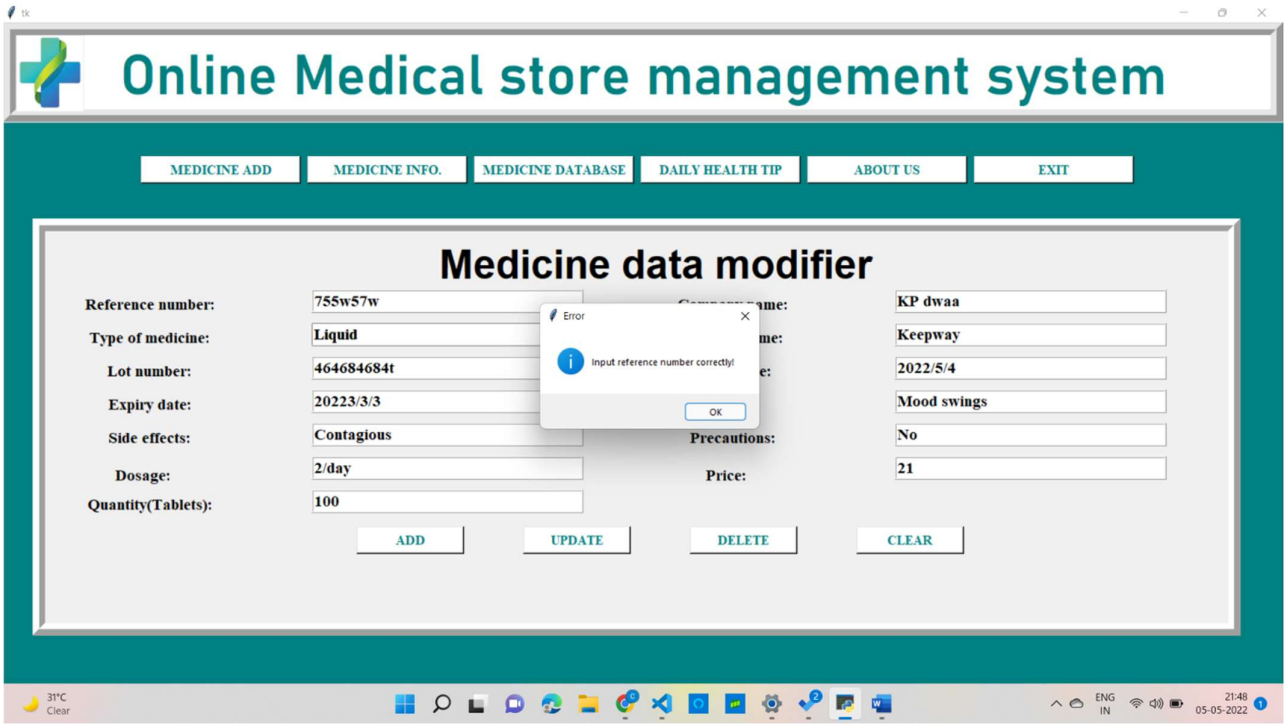
---

Here are the options that one can enter while adding the type of medicine that is going to be added in the database:

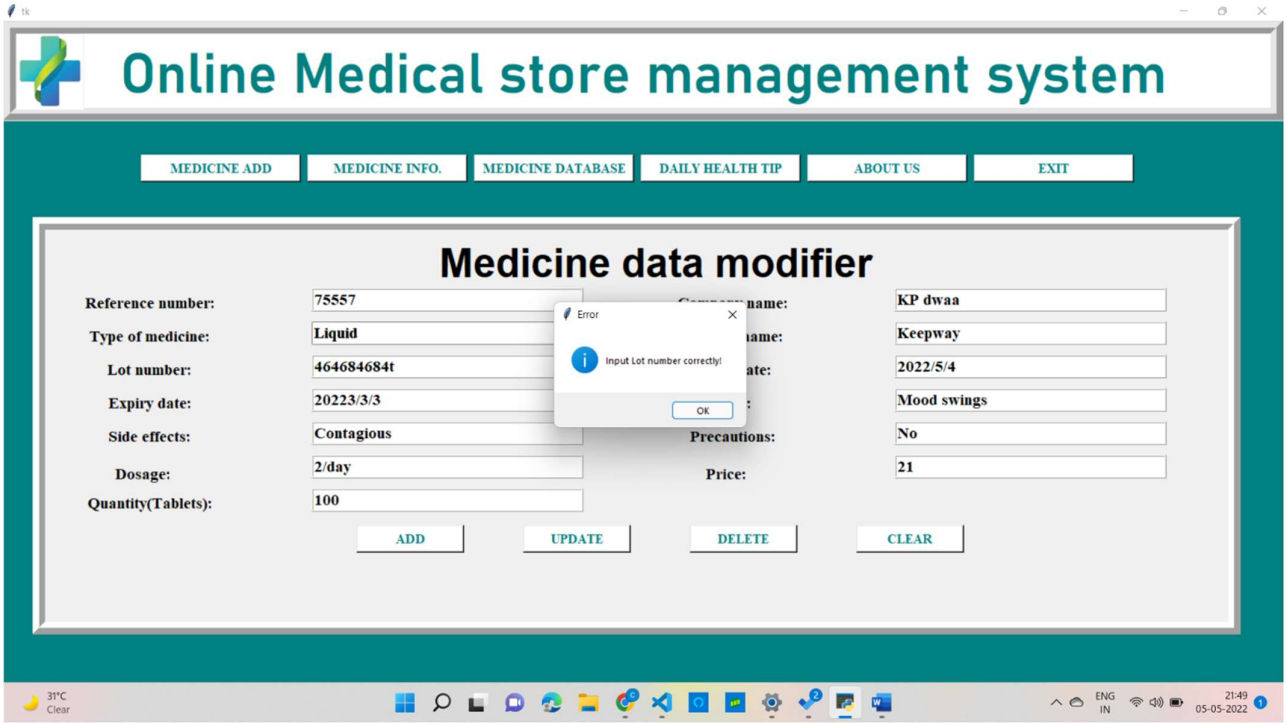


Some error that can be there while inserting medicine in the column are as follows:

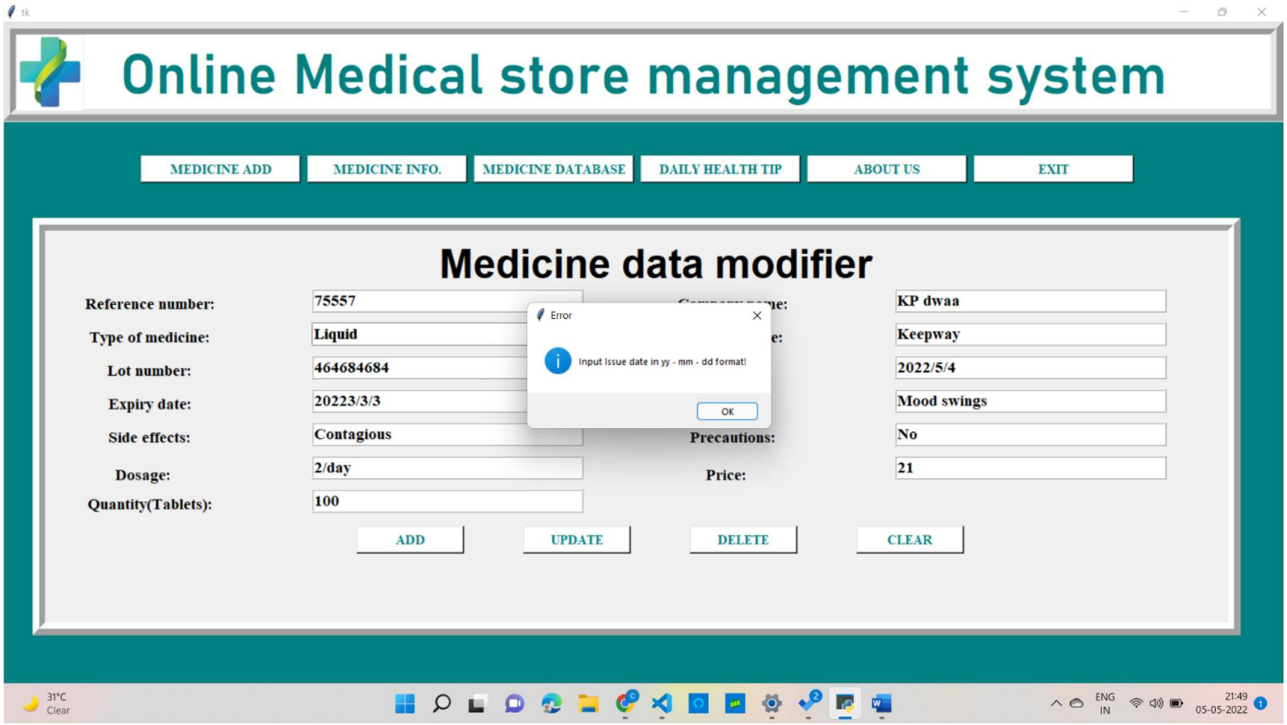
Here is the full of error inputs that we have entered in the blanks and this will display the error message as you entered the wrong inputs.



Here, you can see that reference number is written here in the format which is having the text along with the numbers. But as per the specified text blank format, it was mentioned that reference number should contain the text only in the integer format. Hence, we can see the above-mentioned error message will be displayed on the screen.

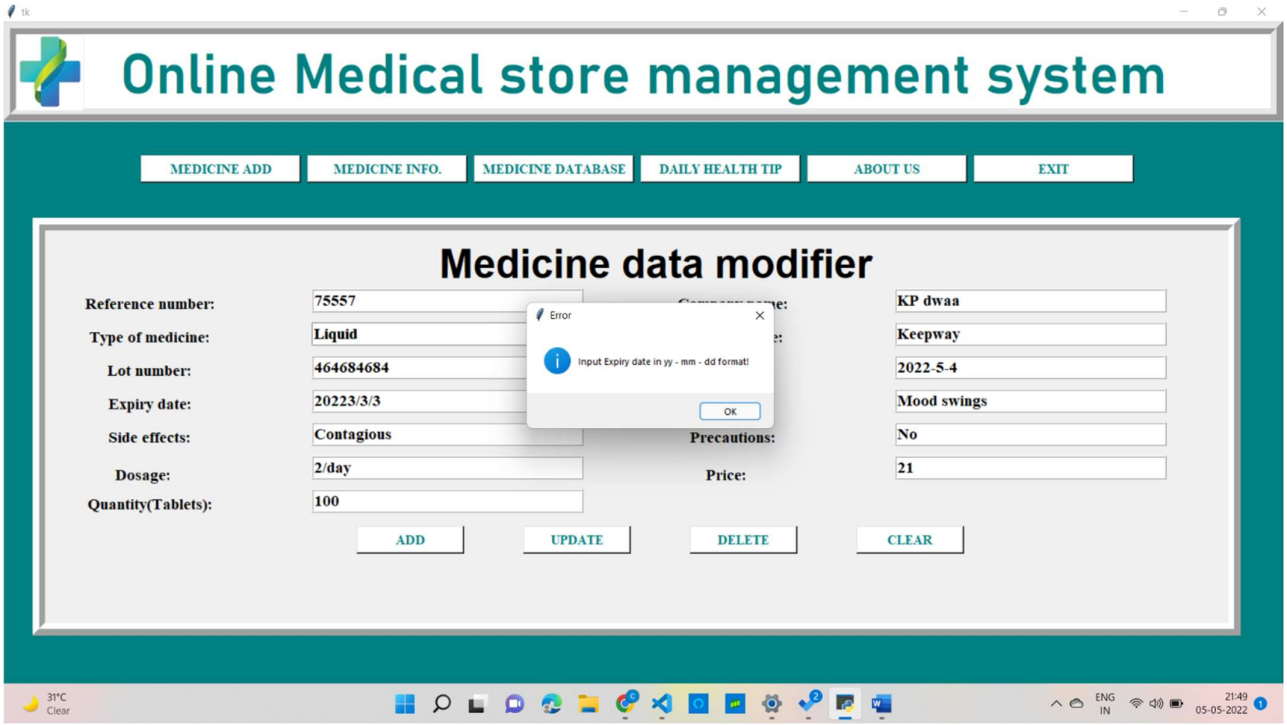


Here, you can see that lot number is written here in the format which is having the text along with the numbers. But as per the specified text blank format, it was mentioned that lot number should contain the text only in the integer format. Hence, we can see the above-mentioned error message will be displayed on the screen.



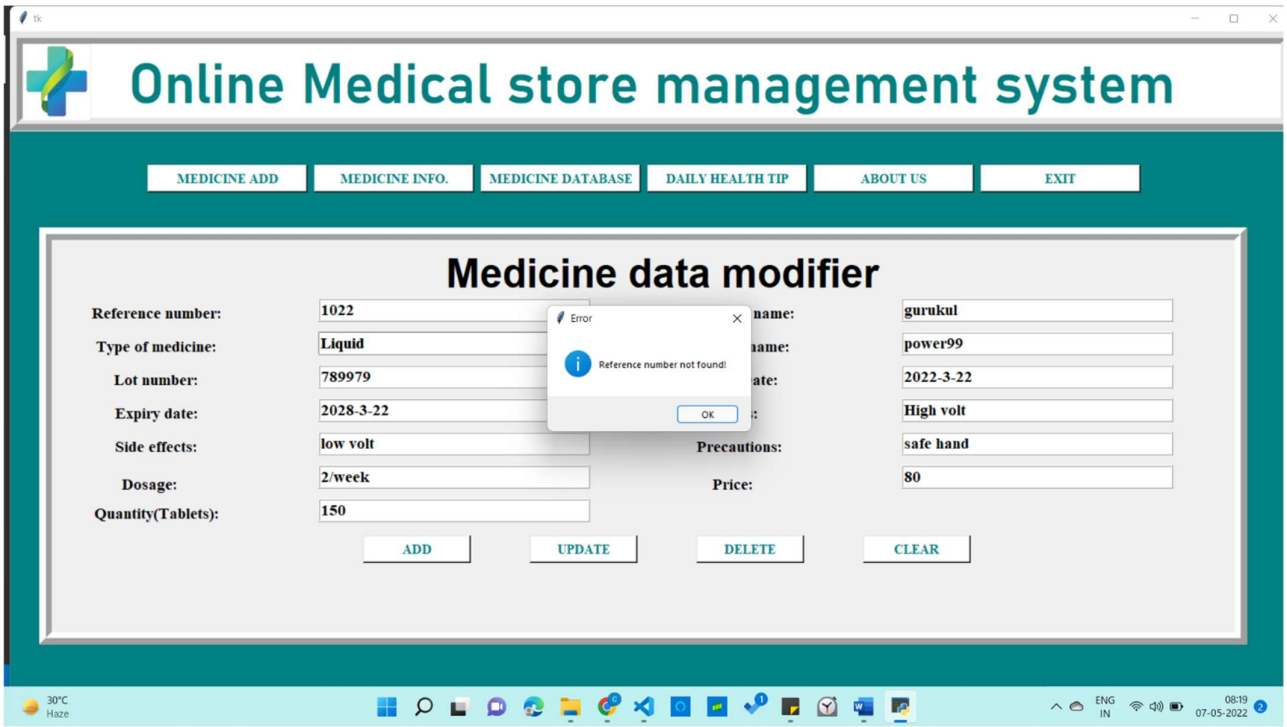
Here, you can see that issue date is written here in the format which is having the text along with the numbers. But as per the specified text blank format, it was mentioned that issue date should contain the text only in the integer format. Hence, we can see the above-mentioned error message will be displayed

on the screen.



Here, you can see that expiry date is written here in the format which is having the text along with the numbers. But as per the specified text blank format, it was mentioned that expiry date should contain the text only in the integer format. Hence, we can see the above-mentioned error message will be displayed on the screen.

While updating any medicine details, you may enter the medicine reference number that is not present in your database, so there will be an error that will pop like "Reference number not found"



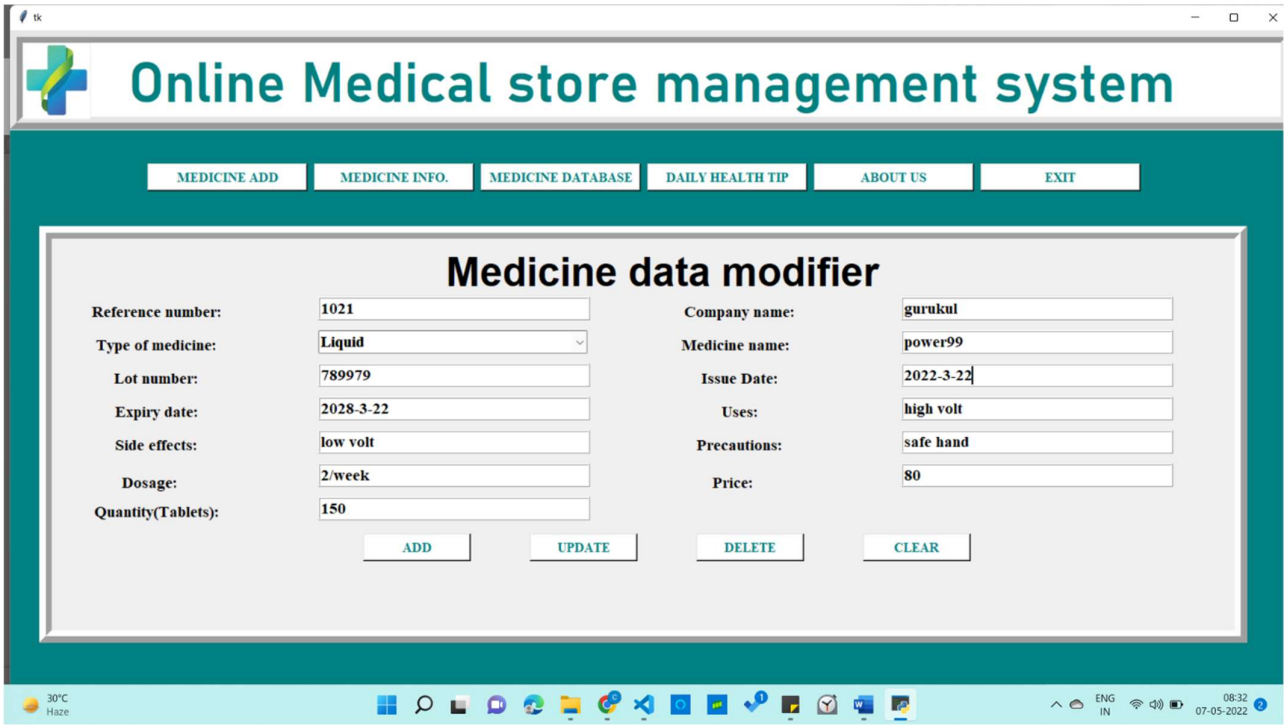
Part 2: Updating the medicine details and all.

Here, now you need to update any information about the medicine, the change in information about the project may include:

- Updating medicine company from where you are buying the medicine.
- Changing the type of medicine.
- To change the lot number according to the new stock.
- To change the issue and expiry date of the medicine as per the new stock that is now available in the store.
- To change the price of the medicine.
- To update the quantity of medicine left in store.

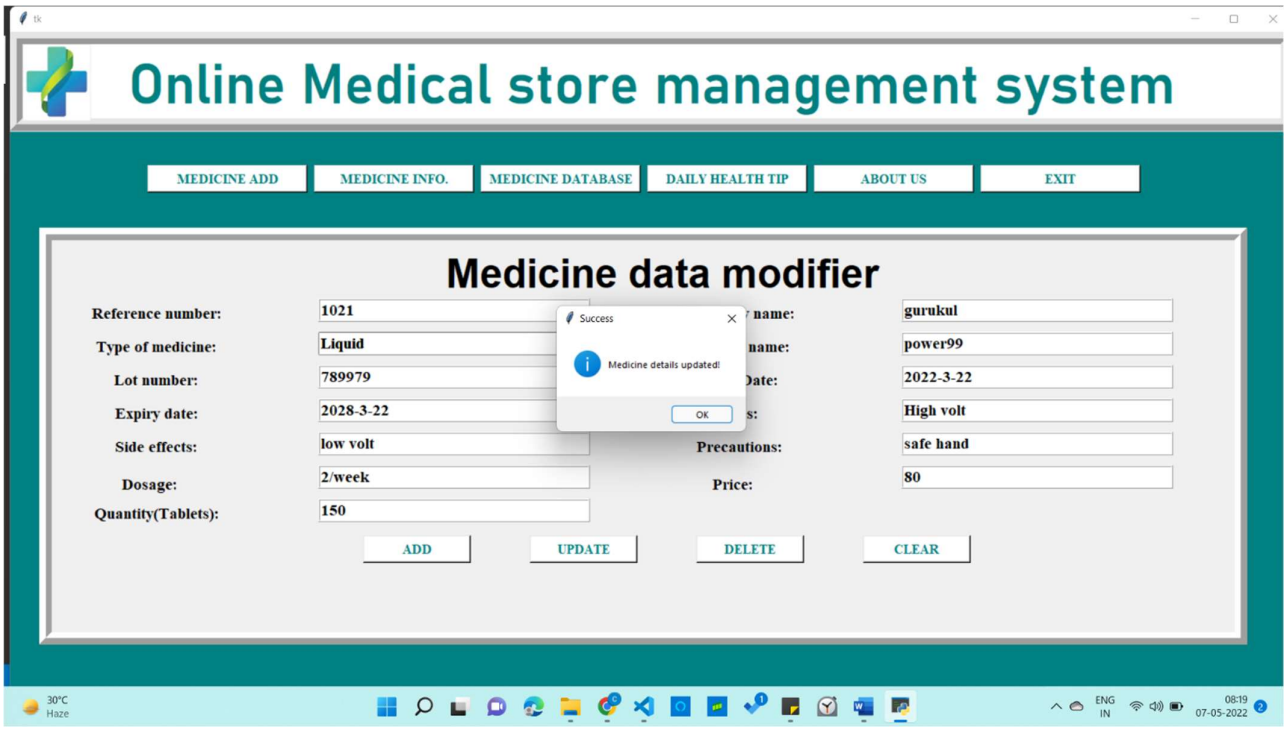
If you are willing to perform any of the task that are mentioned above, so you can input the details of the medicine in the Medicine data modifier and kindly get it updates.7

Now click on update button so that it will be updates as for now.

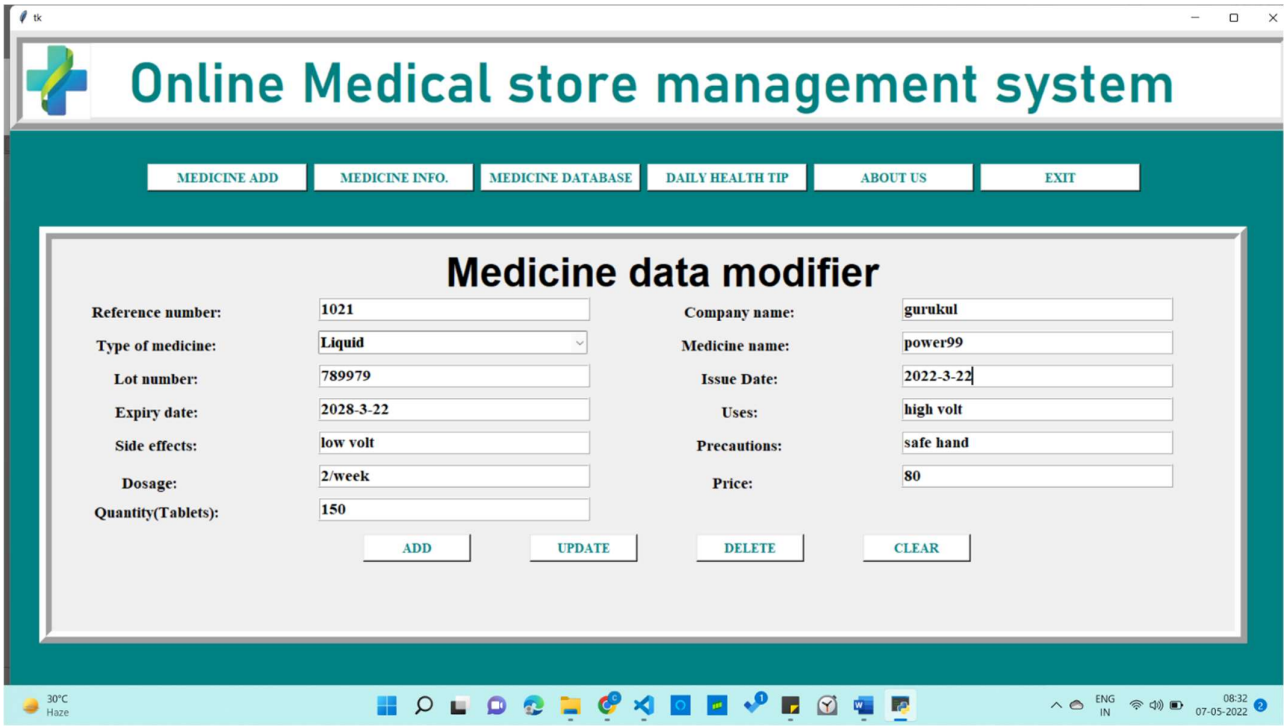


Here you can easily observe the change in

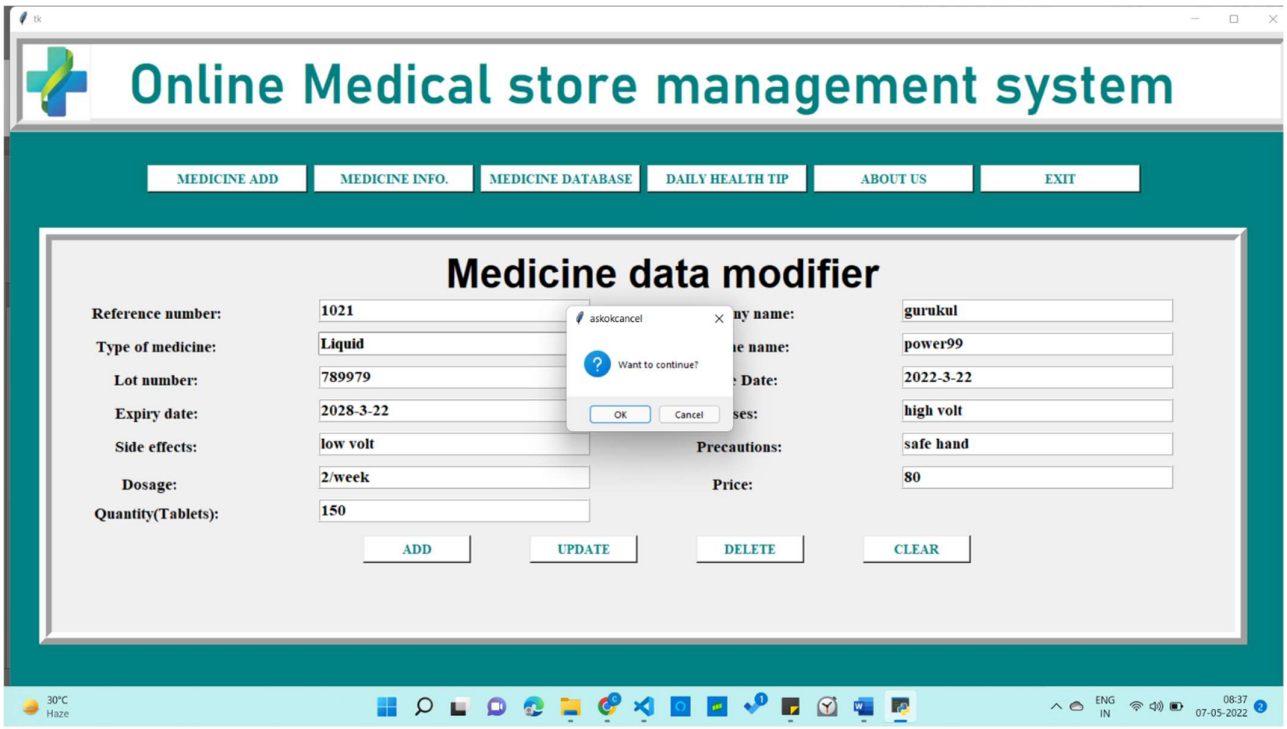
- Uses
- Side effects
- Precautions
- Price
- Quantity



3. Next option is to delete the medicine so you need to write the details of the medicine and click on the delete message.

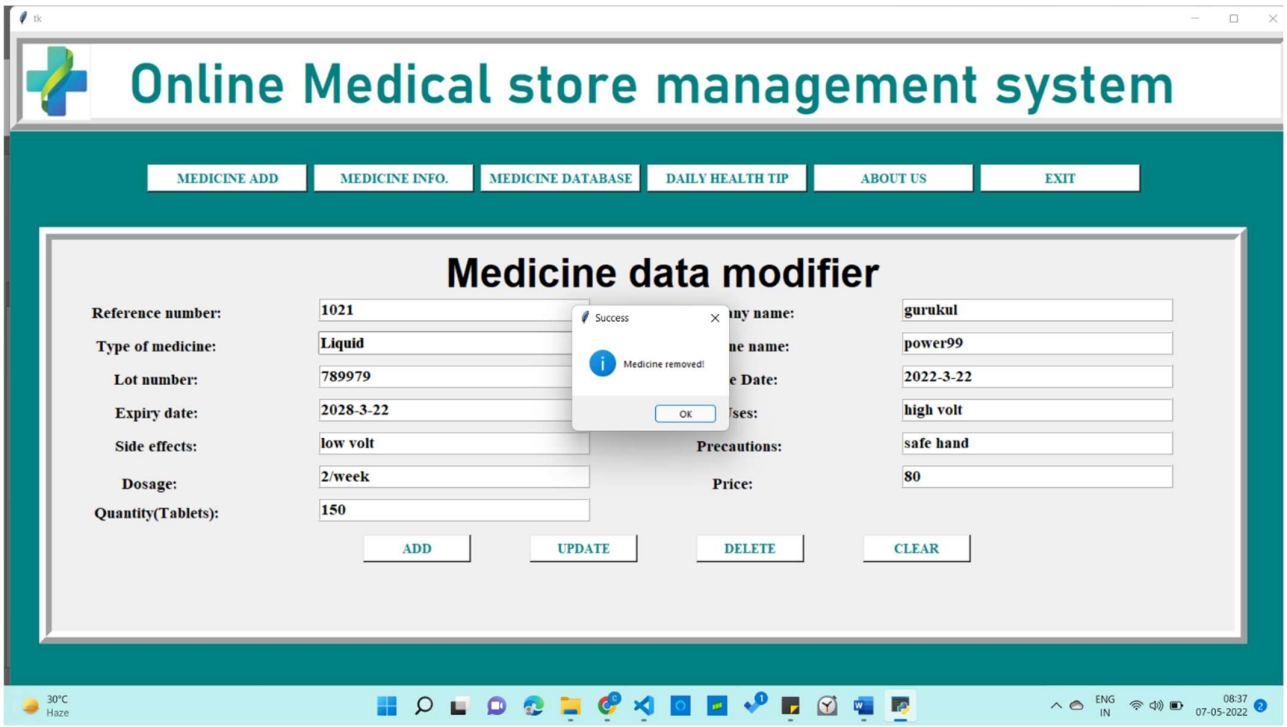


Now, you need to click on the delete button, so the database related to this will be removed .



This time, before deleting the medicine, it will ask for confirmation that we surely want to delete this message from the database. It will show up message: Do you ant to continue?

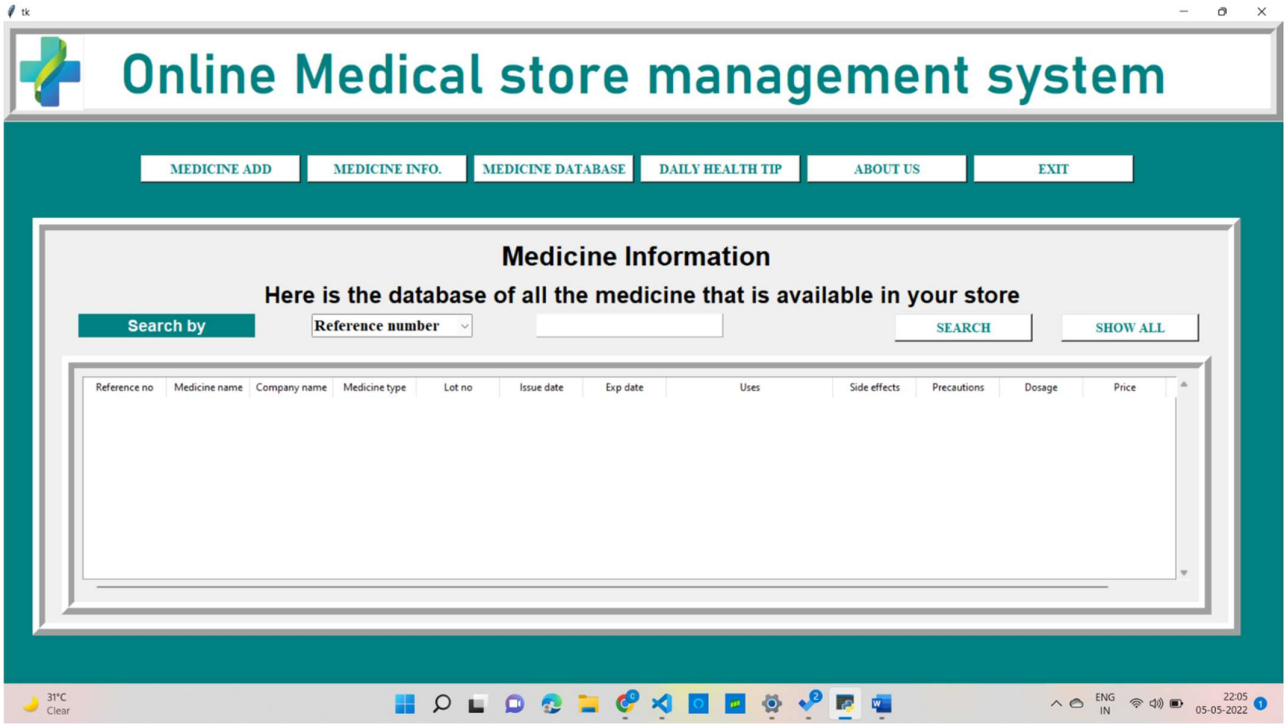
Press OK for the confirmation.



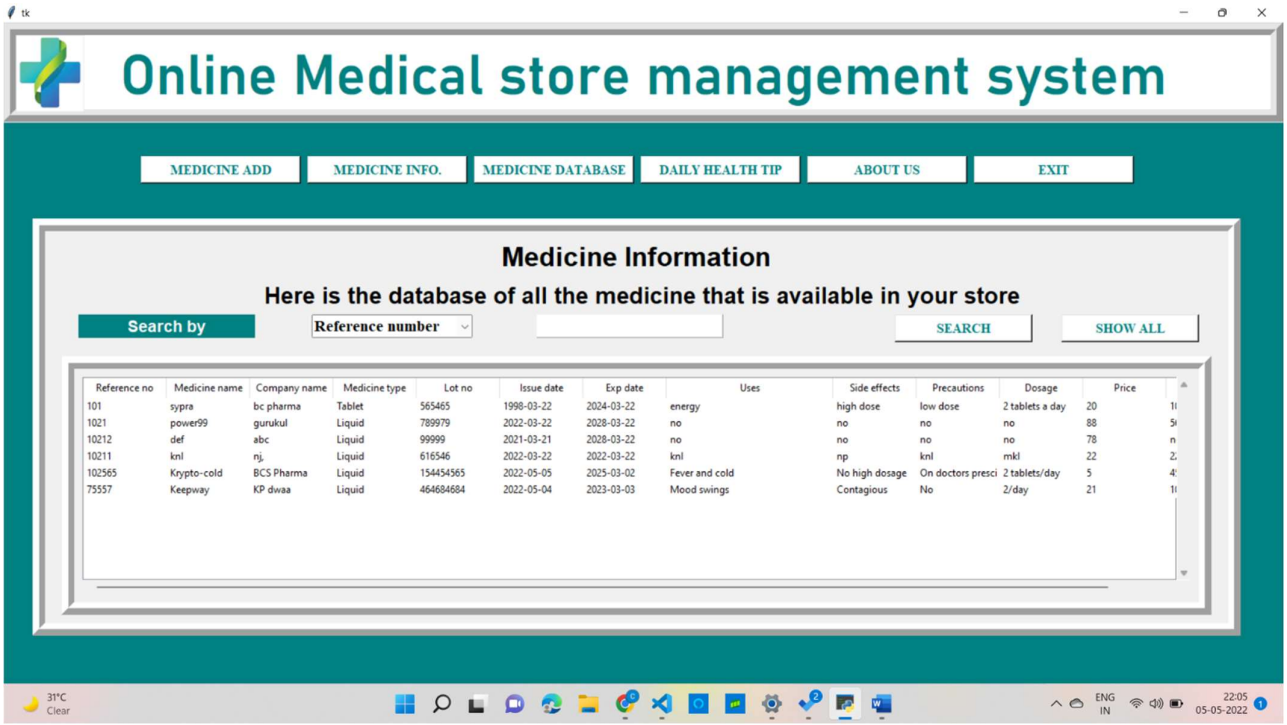
Now your medicine will be removed from the database.

Next button is medicine database:

Here, you can search for the database and also look for the complete database at once.



Assume that you want to check the complete database for the medicine so on the above shown page, you need to click on the show all button to view the complete database.



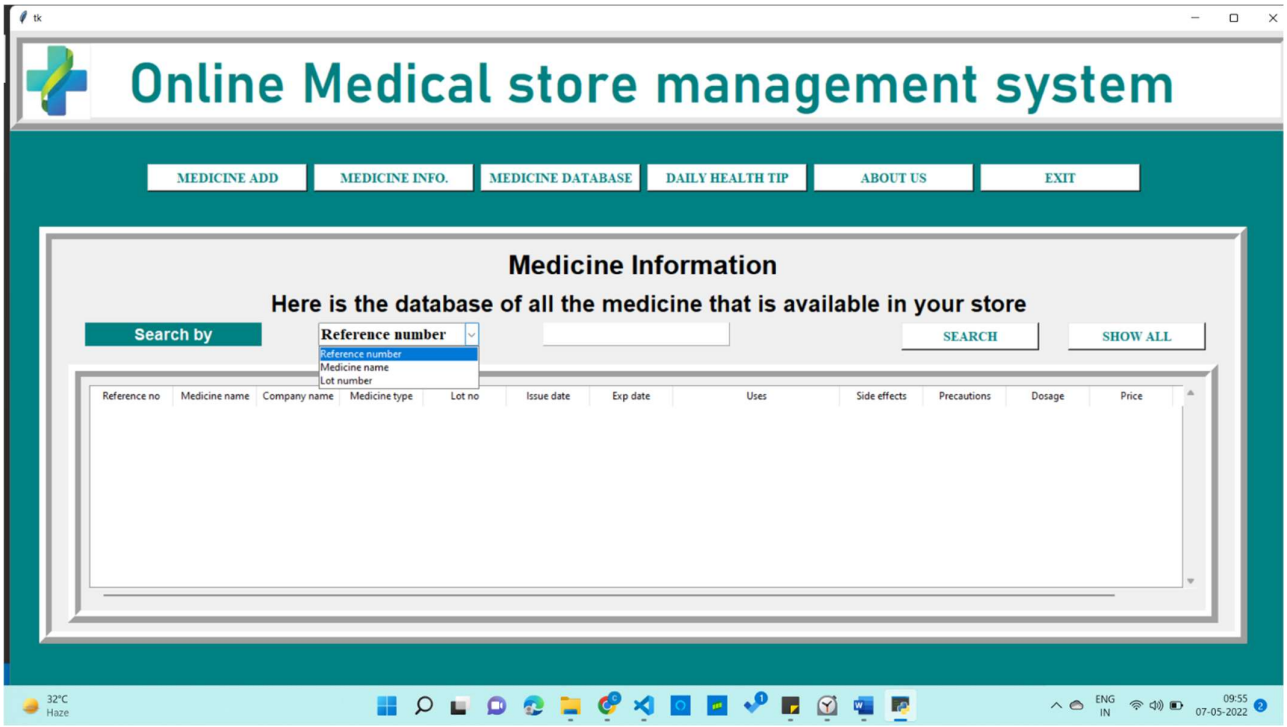
Here,

This is the complete database of our online medical store management system.

The other option we can try is using the search button for custom search in the database.

We can search the content in the database based on the following options:

- Reference number
- Medicine name
- Lot number



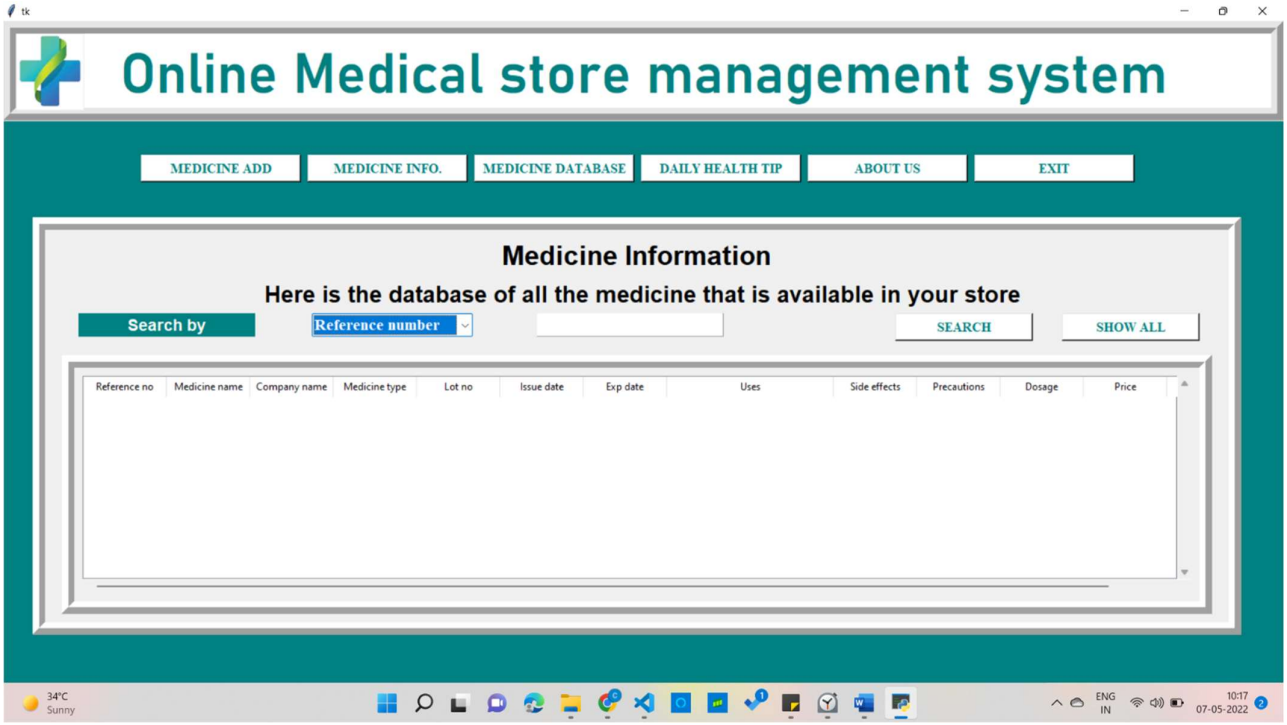
Here is the above given illustration of the management system. The basic input for the choice of search by option is reference number. These all three are chosen as basic option for searching a medicine because:

- **Reference number:** Reference number is the primary key in one of the tables mentioned in the online medical store management system. And in the second table of the database, it is used as the foreign key so we can say that it can uniquely identify the medicine.
- **Medicine name:** Medicine name is the key associated with the reference key (primary key) in one of the tables mentioned in the online medical store management system. And in the second table of the database, it is used as the foreign key so we can say that it can uniquely identify the medicine.
- **Lot number:** Lot number is the primary key if one of the tables mentioned in the online medical store management system. So, we can say that it can uniquely identify the medicine.

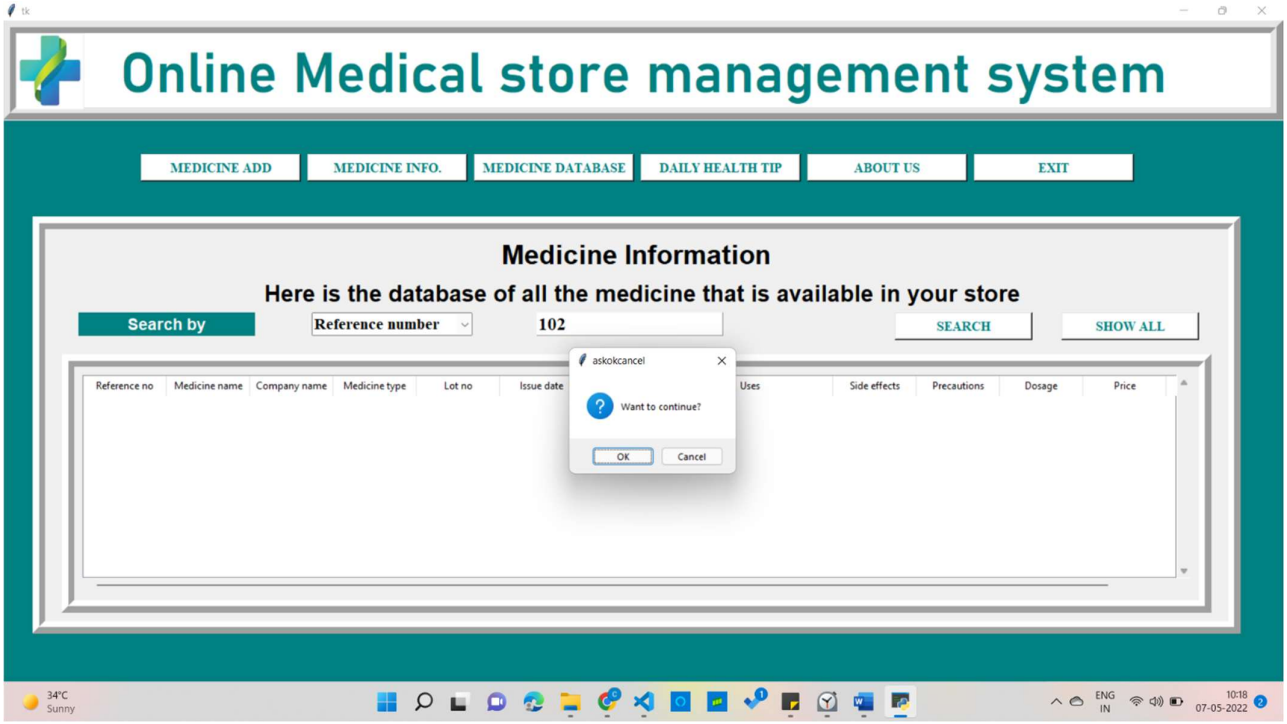
Case 1:

Searching by reference number:

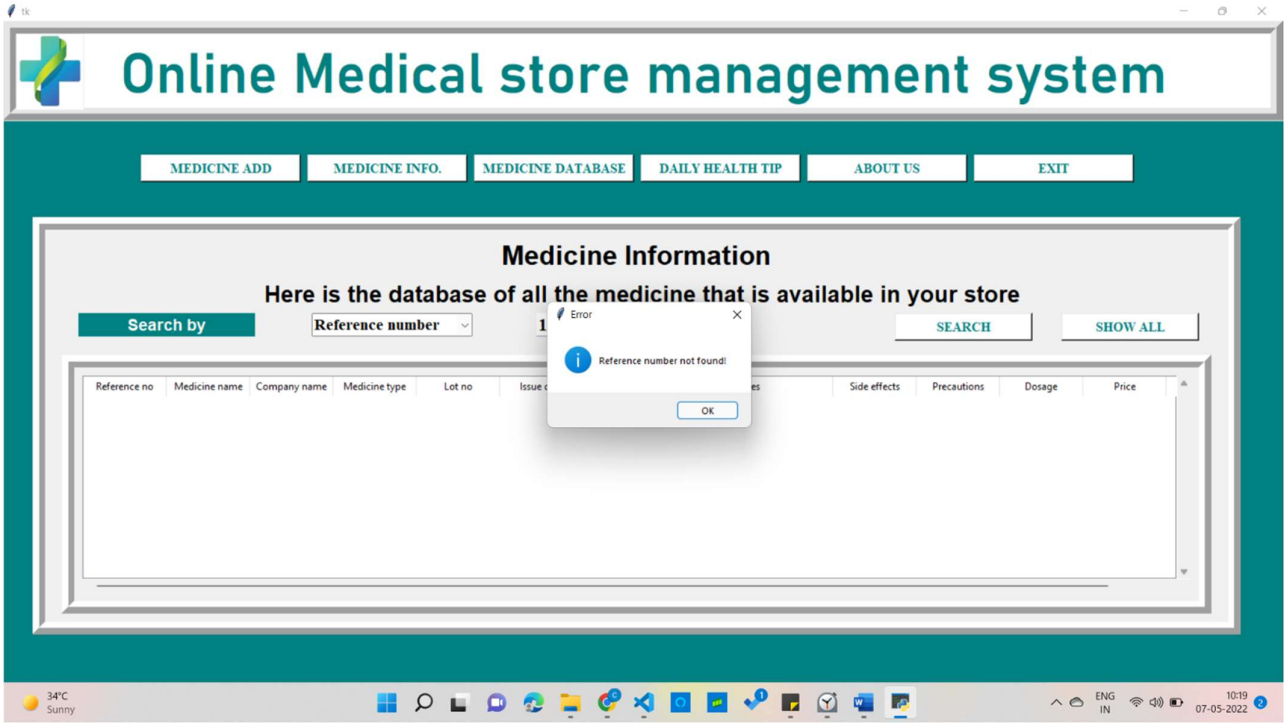
1. Choose reference number



2. Enter any reference number:
   - For instance, we choose 102 as the reference number.
   - So, first it will ask for whether to continue or not?



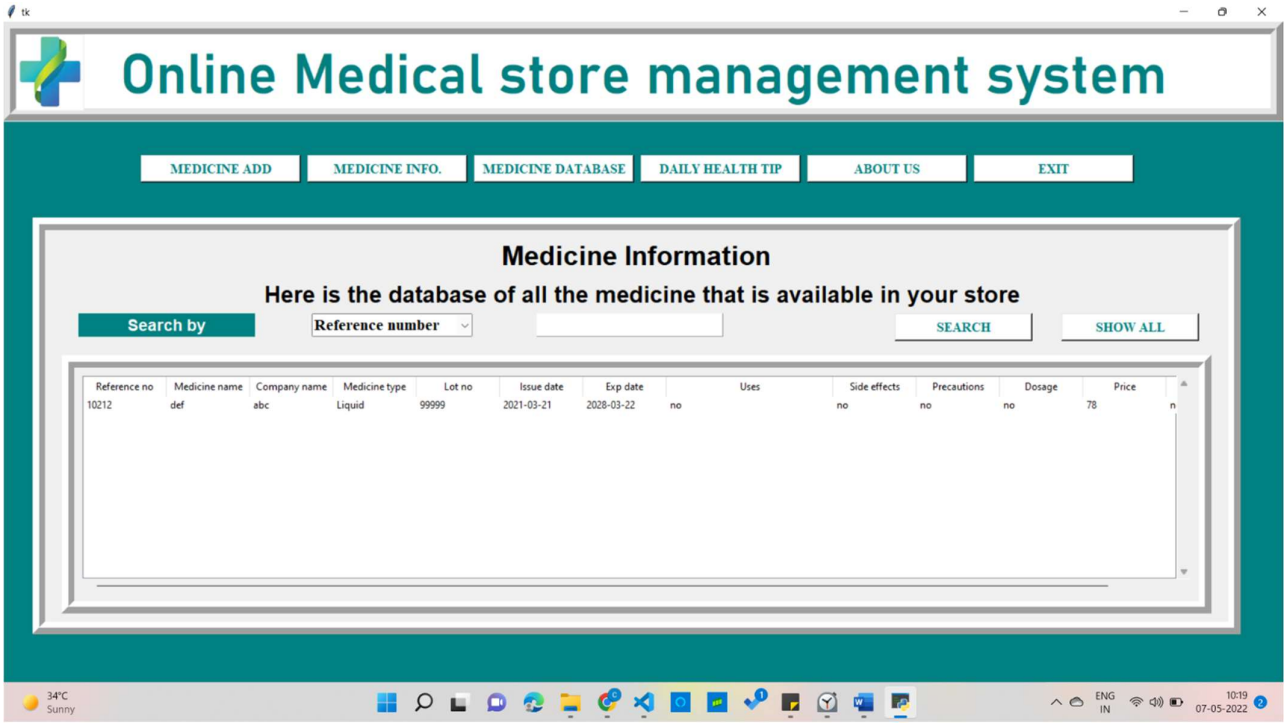
   - Click yes.
   - Then the result will be shown as per the query
   - 102 is not present as the reference number in the database so the following message is shown in the message box: Reference number not found!



The above message shows the error message that will be displayed while searching for the medicine with reference number 102 in the database.

Now, we will try for searching with the reference number 10212.

Then it will again ask: Do you want to continue? Press yes.

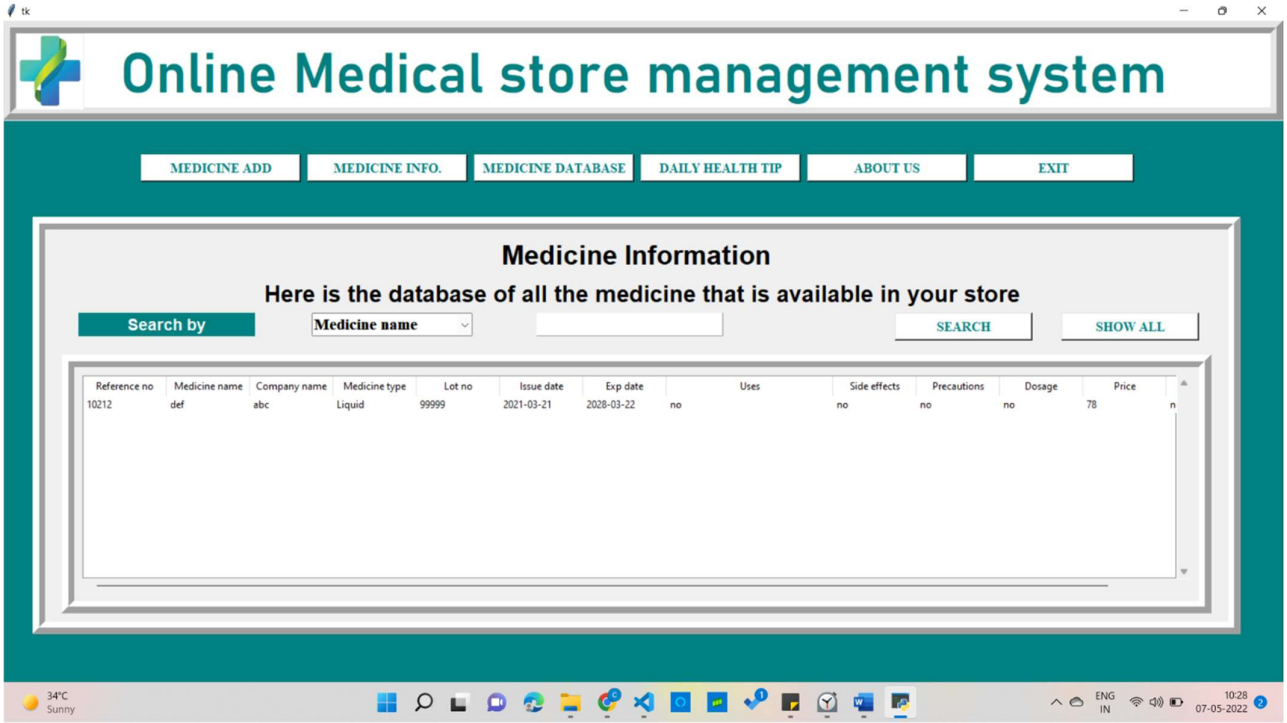


The result is shown as shown in the above figure that all data including medicine name, company name, lot number, etc is displayed on the screen

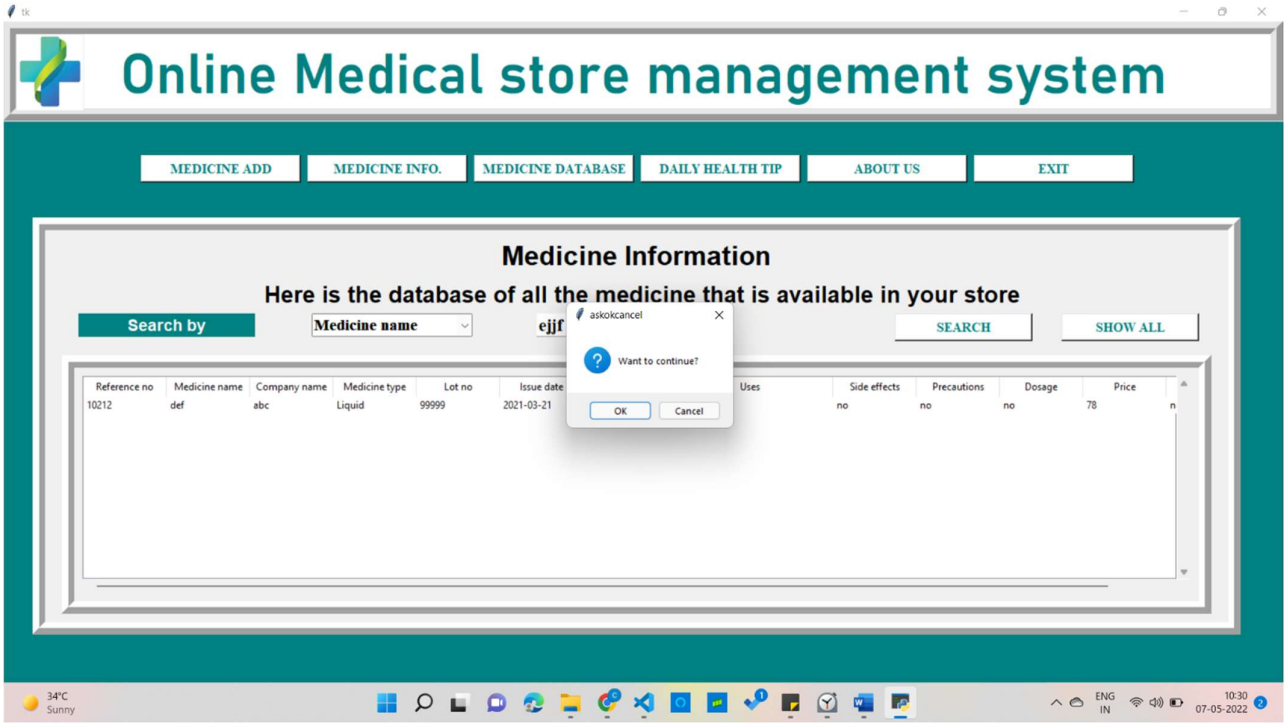
NOTE: Reference number must be an integer only.

Case 2: Search by: Medicine name

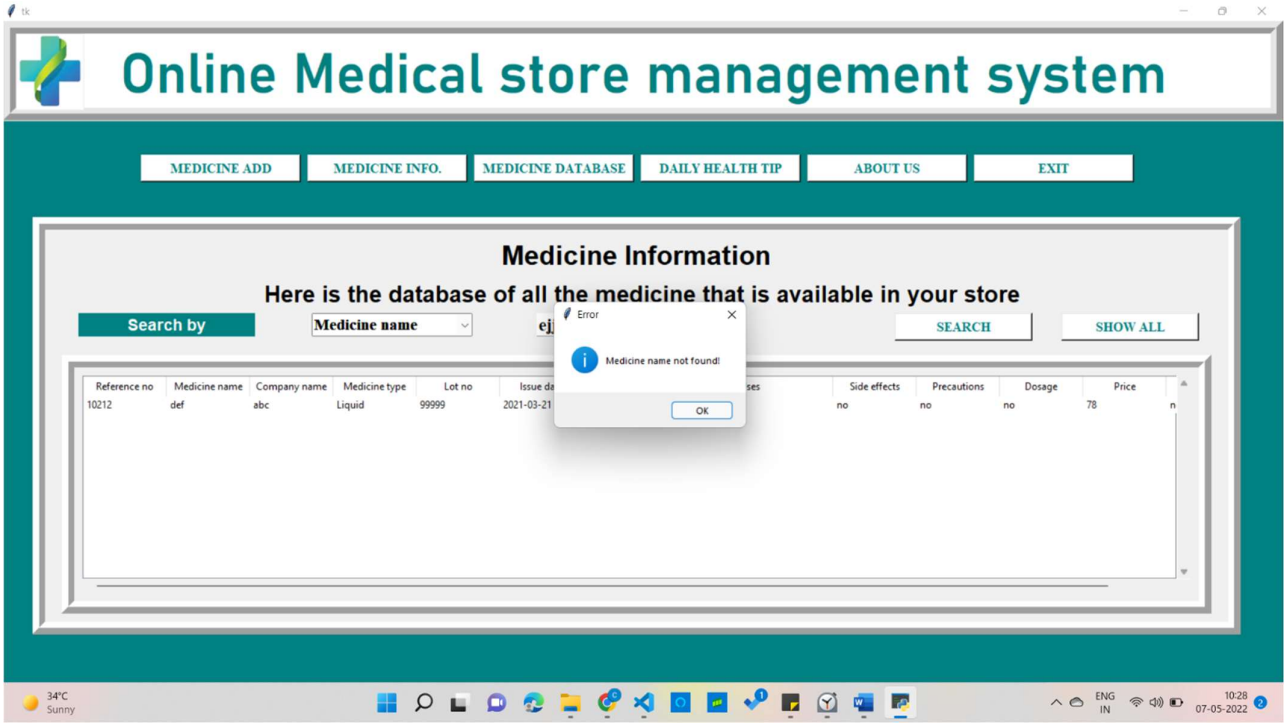
1. Choose medicine name



2. Enter any reference number:
   - For instance, we choose ejjf as the reference number. So, first it will ask for whether to continue or not?



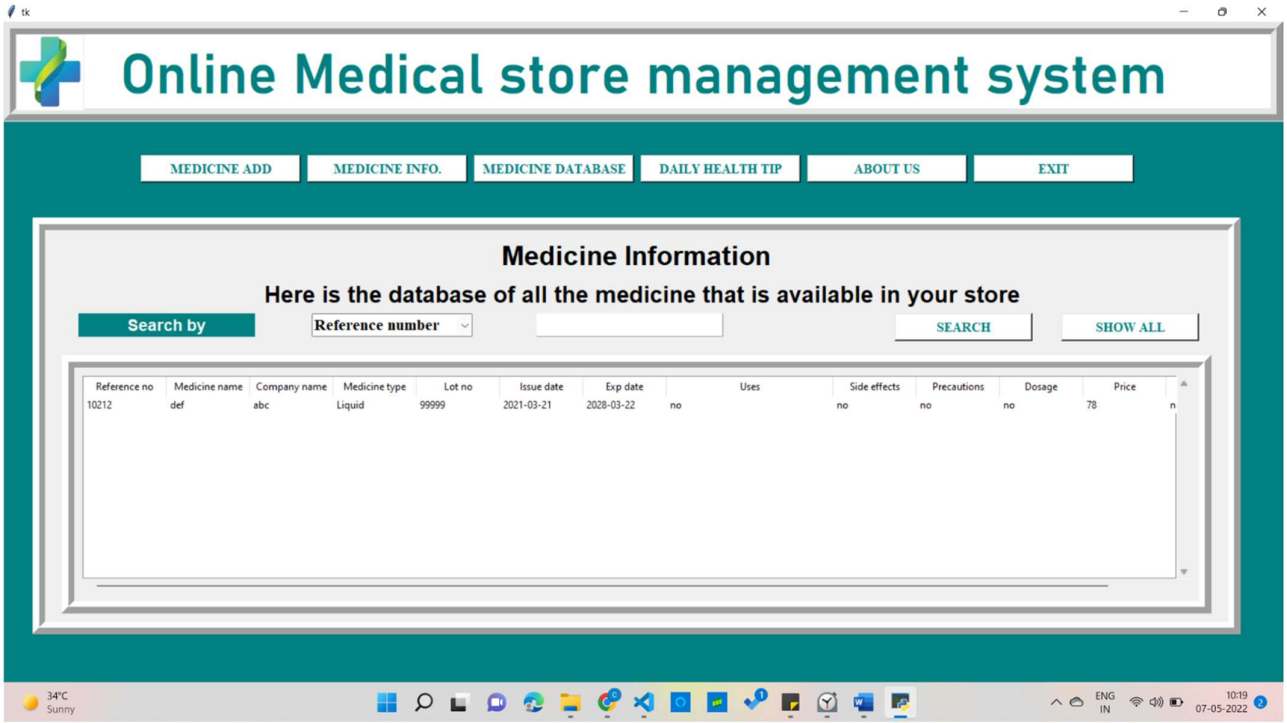
   - Click yes.
   - Then the result will be shown as per the query
   - ejjf is not present as the medicine name in the database so the following message is shown in the message box: Medicine name not found!



The above message shows the error message that will be displayed while searching for the medicine with medicine name ejjf in the database.

Now, we will try for searching with the medicine name def

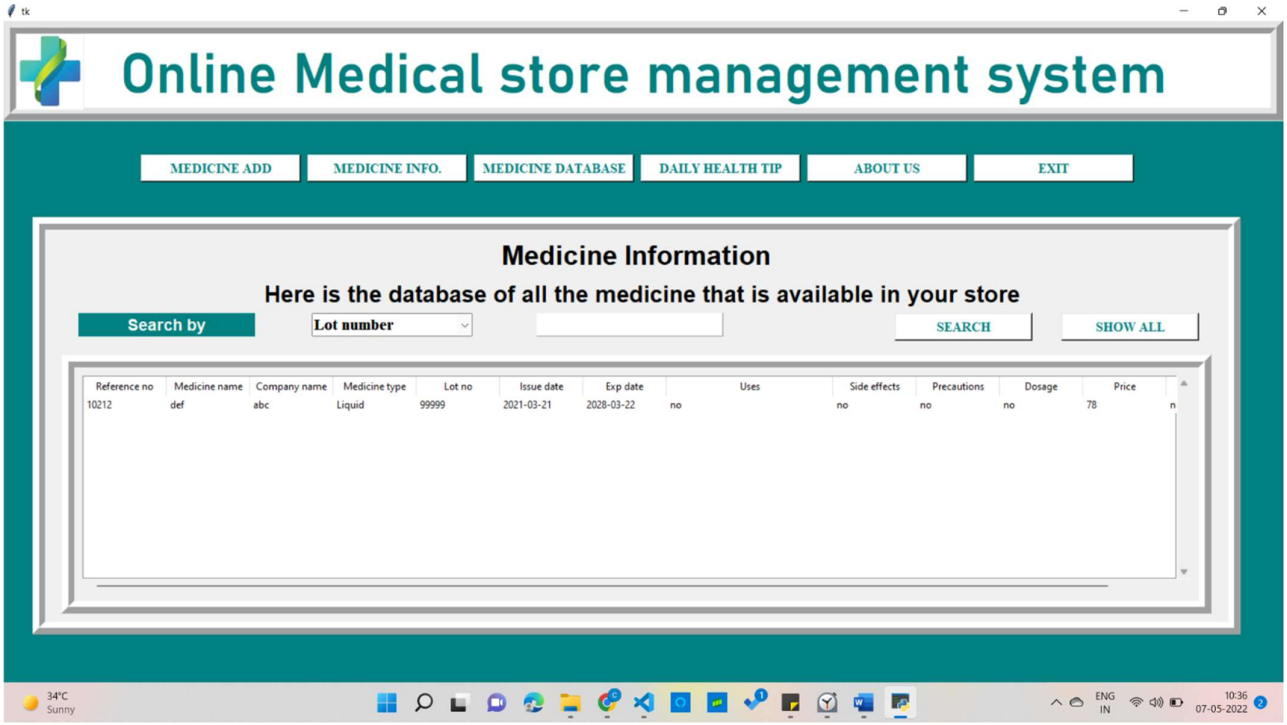
Then it will again ask: Do you want to continue? Press yes.



The result is shown as shown in the above figure that all data including medicine name, company name, lot number, etc is displayed on the screen

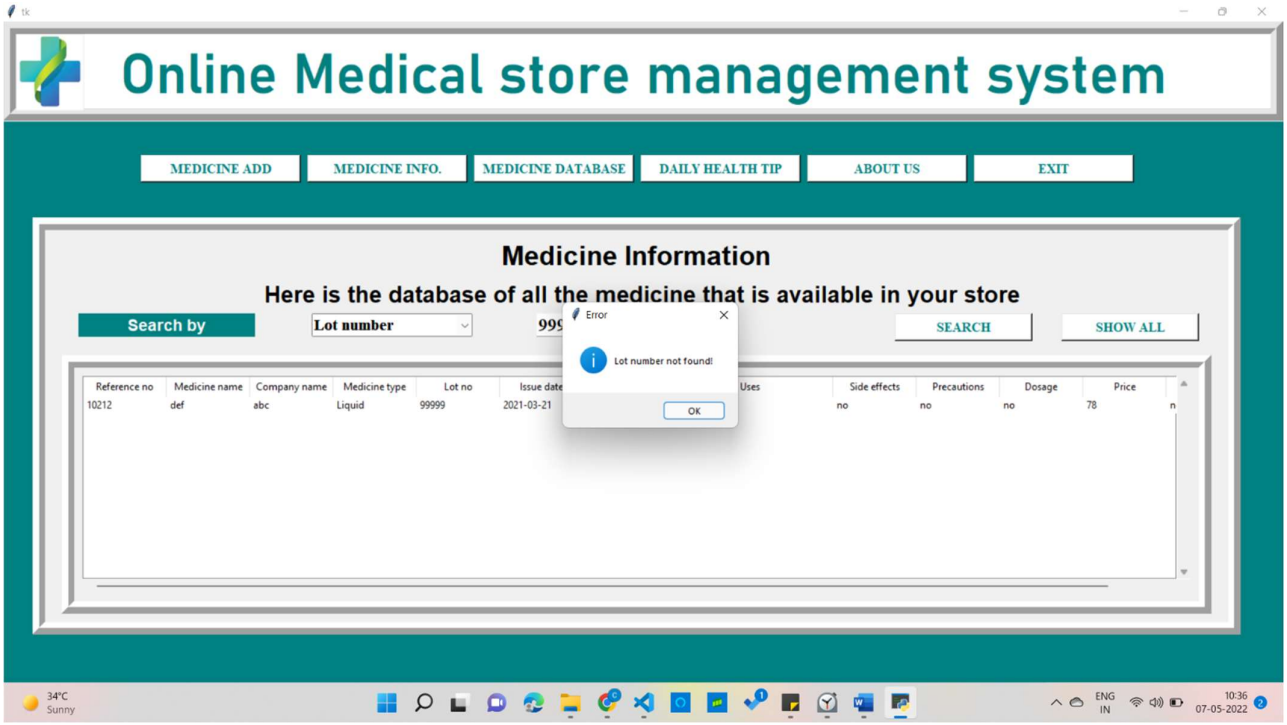
Case 2: Search by: Lot number

1. Choose Lot number



2. Enter any lot number:

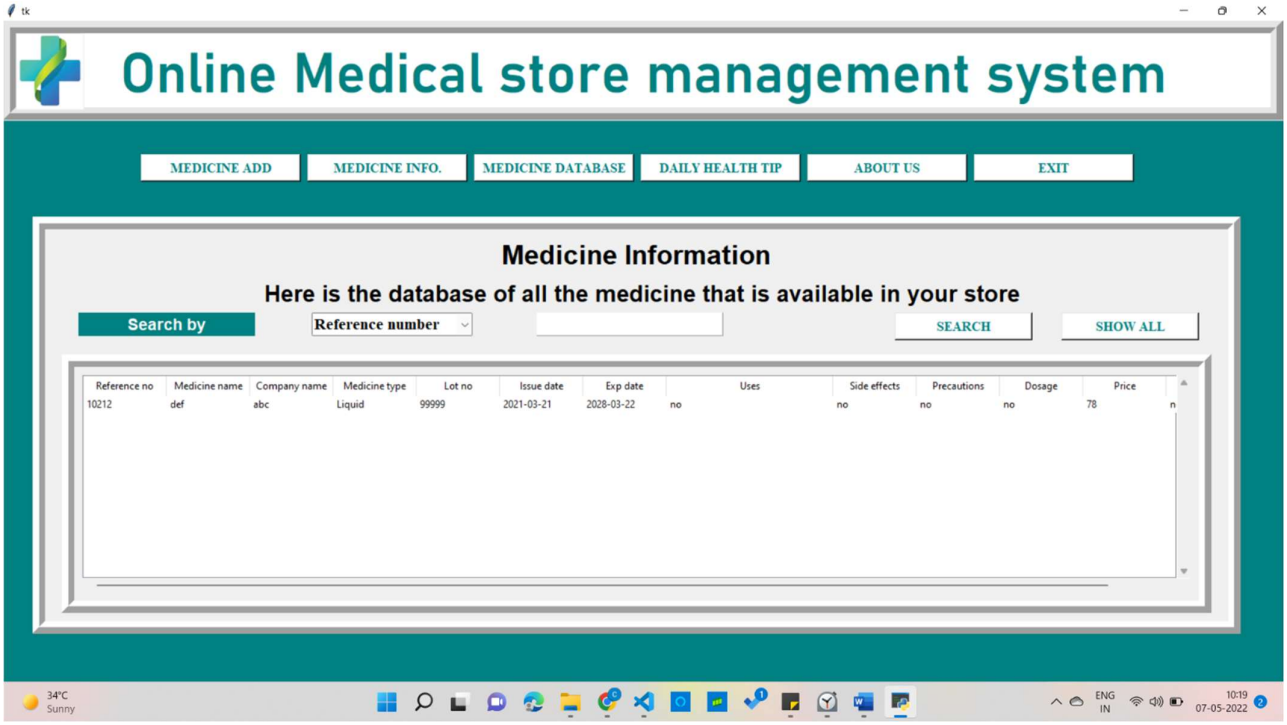
- For instance, we choose 9999 as the lot number. So, first it will ask for whether to continue or not? Click yes. Then the result will be shown as per the query
- 9999 is not present as the Lot number in the database so the following message is shown in the message box: lot number not found!



The above message shows the error message that will be displayed while searching for the medicine with lot number 9999 in the database.

Now, we will try for searching with the lot number 99999

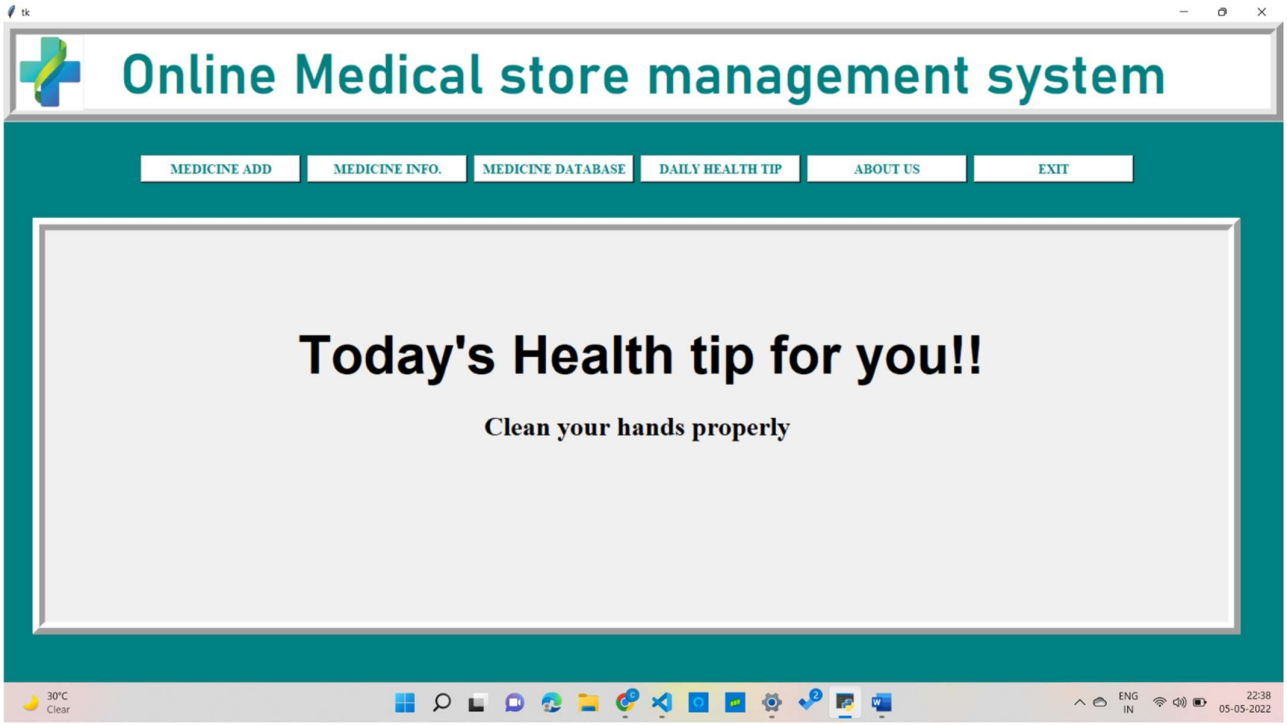
Then it will again ask: Do you want to continue? Press yes.



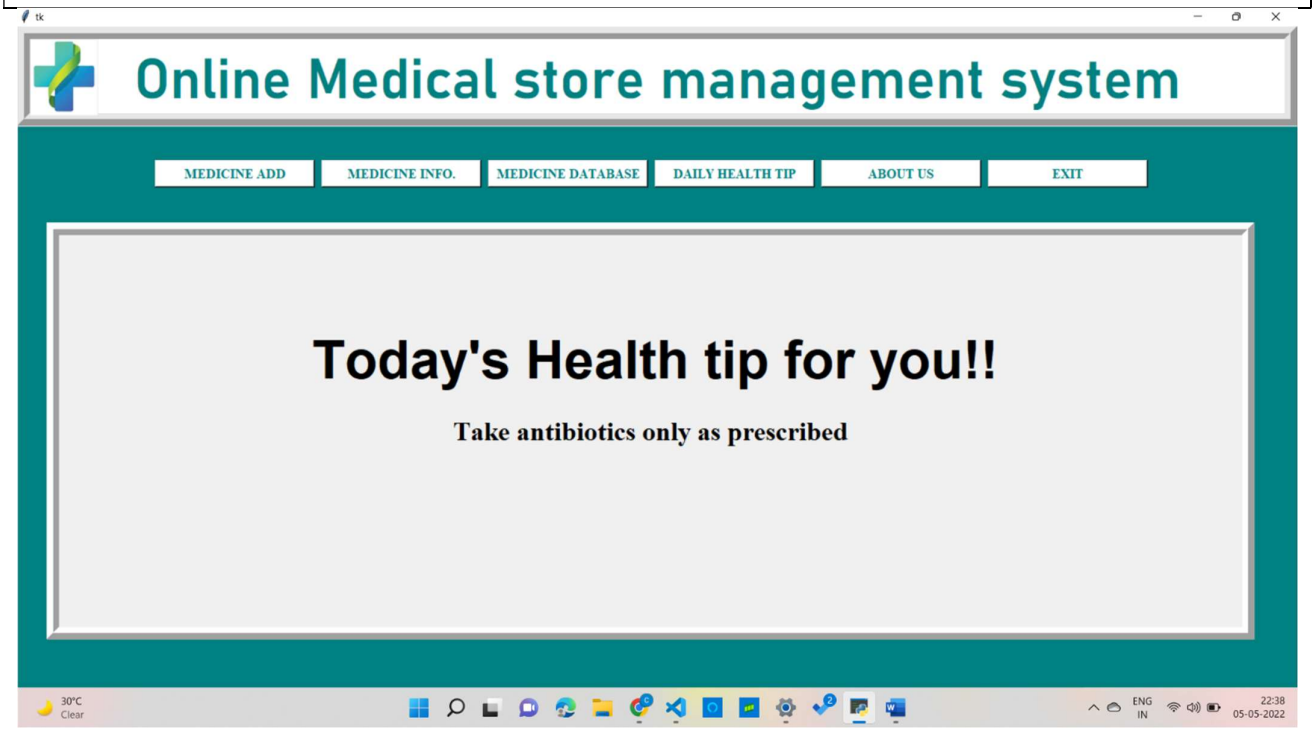
The result is shown as shown in the above figure that all data including medicine name, company name, lot number, etc is displayed on the screen

Next button is the Daily heath tips:

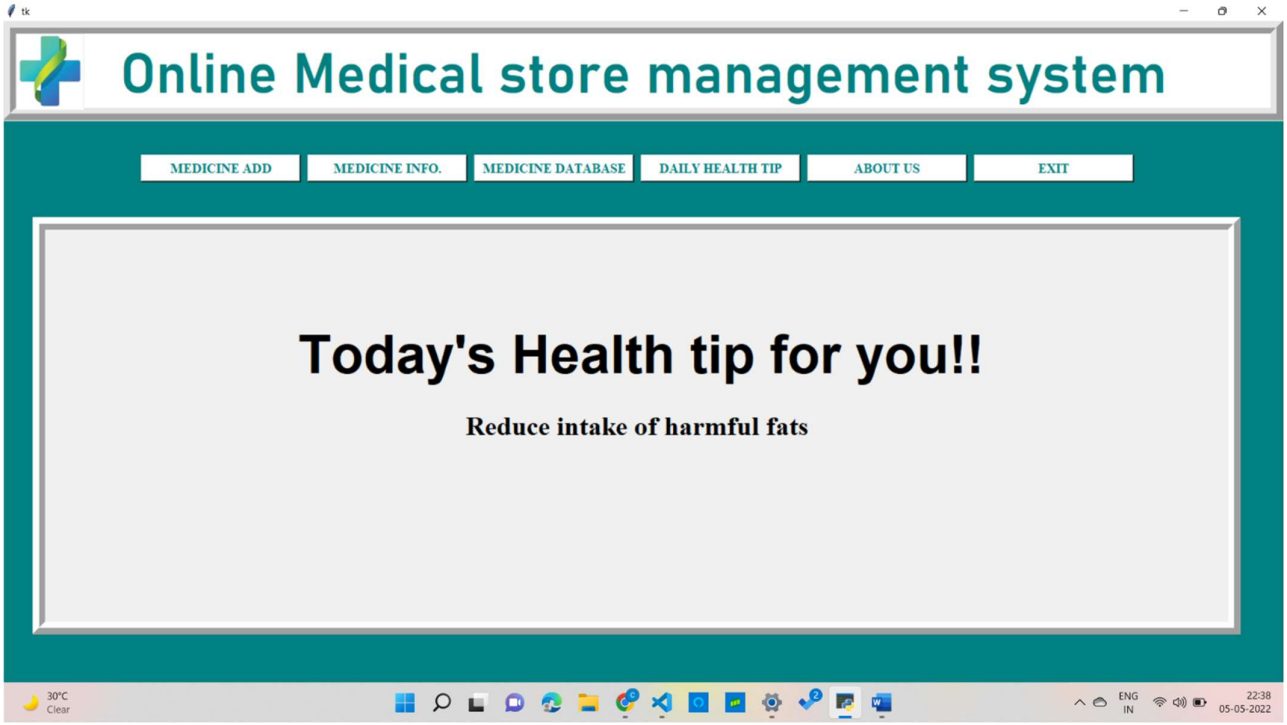
Some of them are as follows:



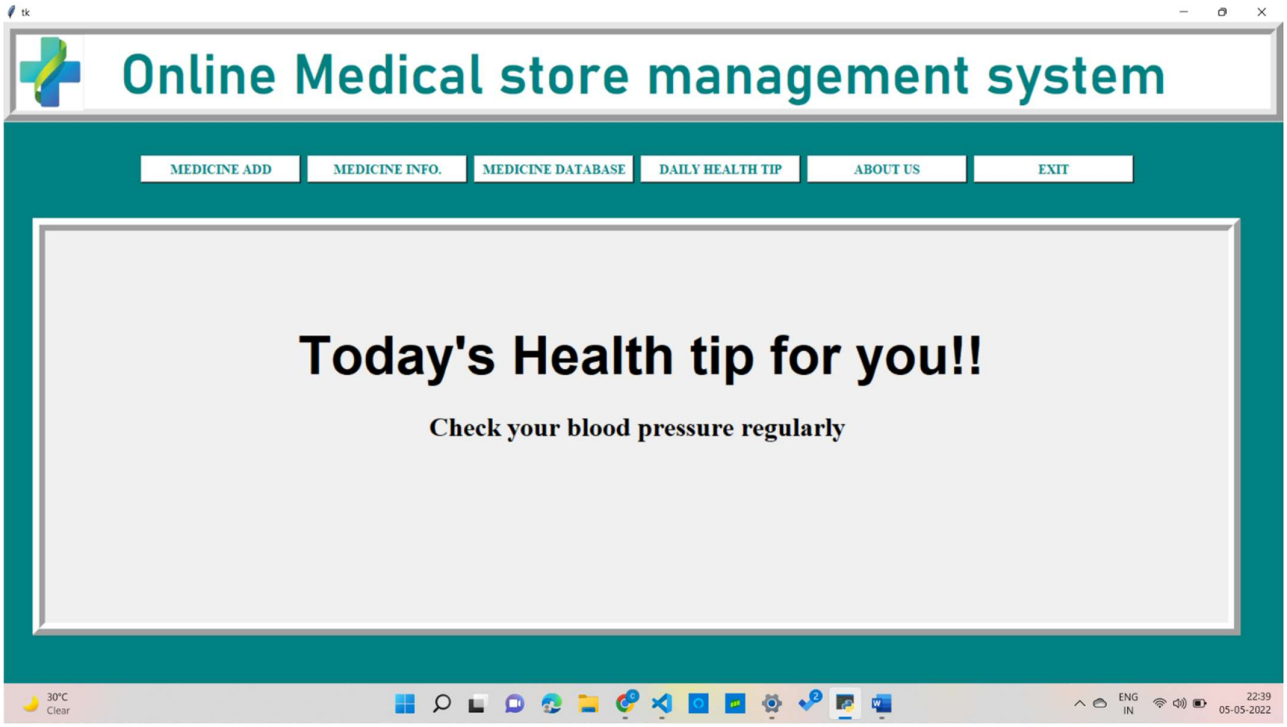
Clean your hands properly



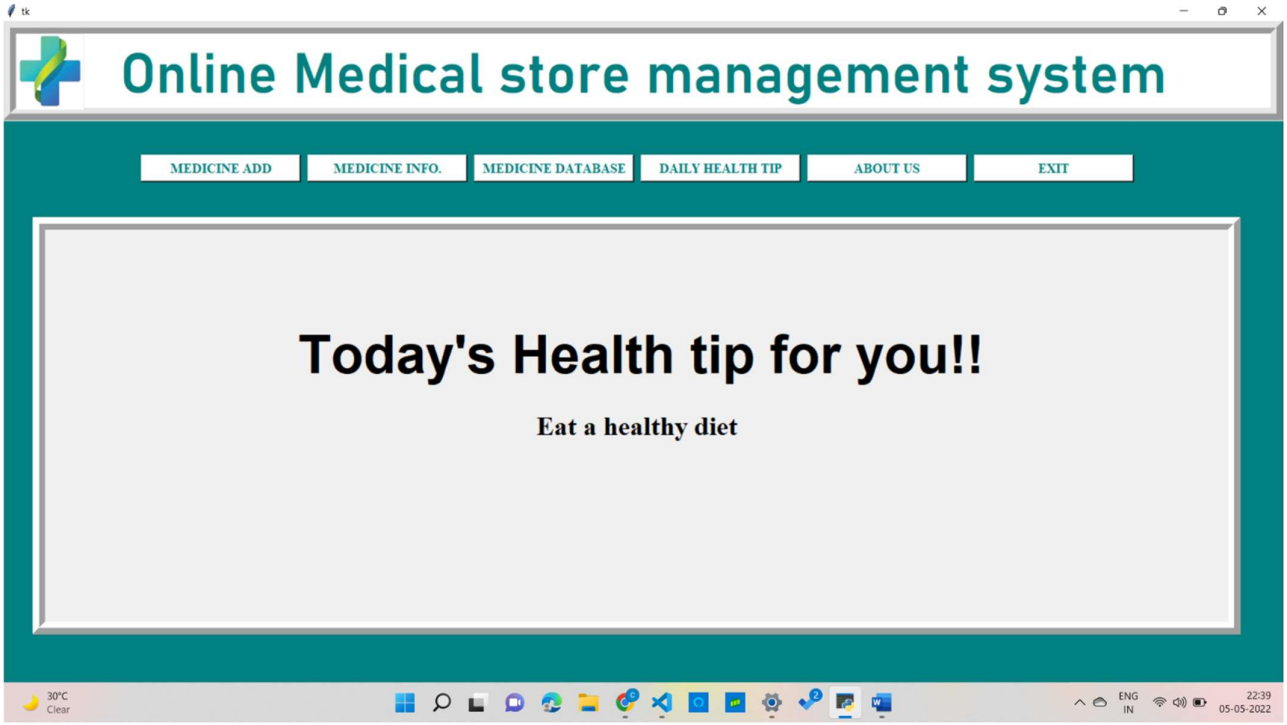
Take antibiotics only as prescribed



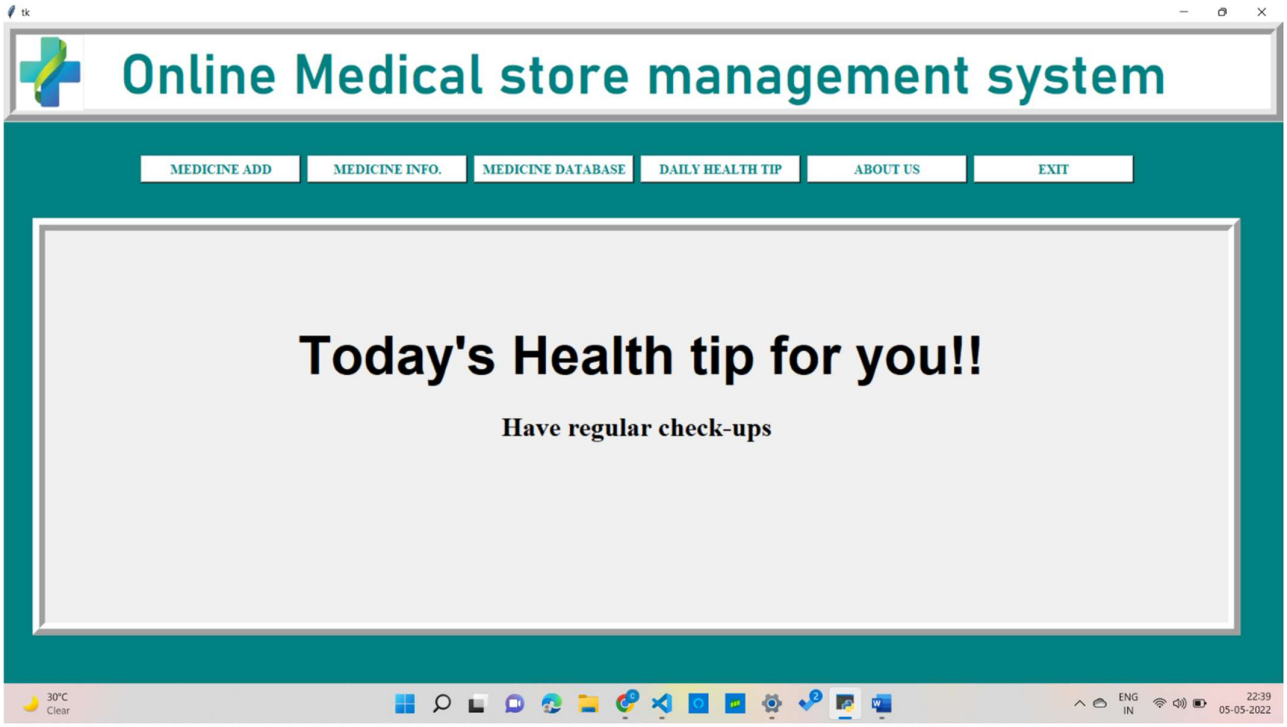
Reduce intakes of harmful fats



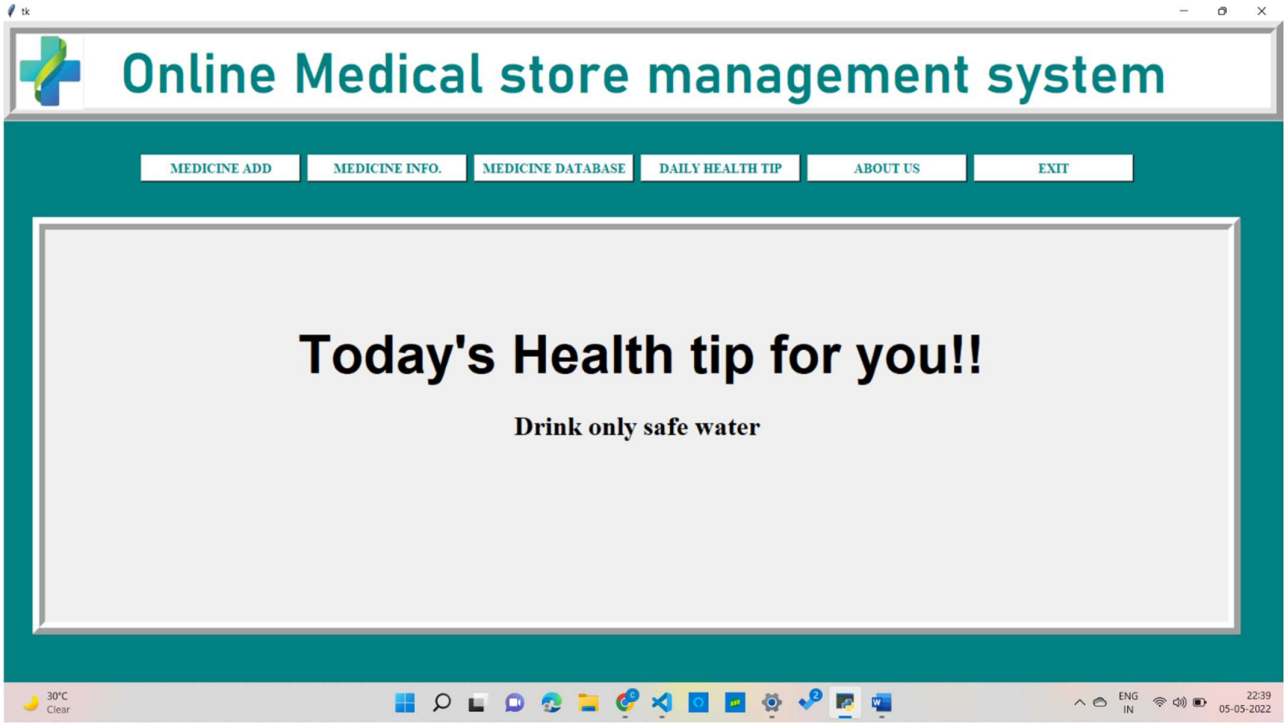
Check your blood pressure regularly



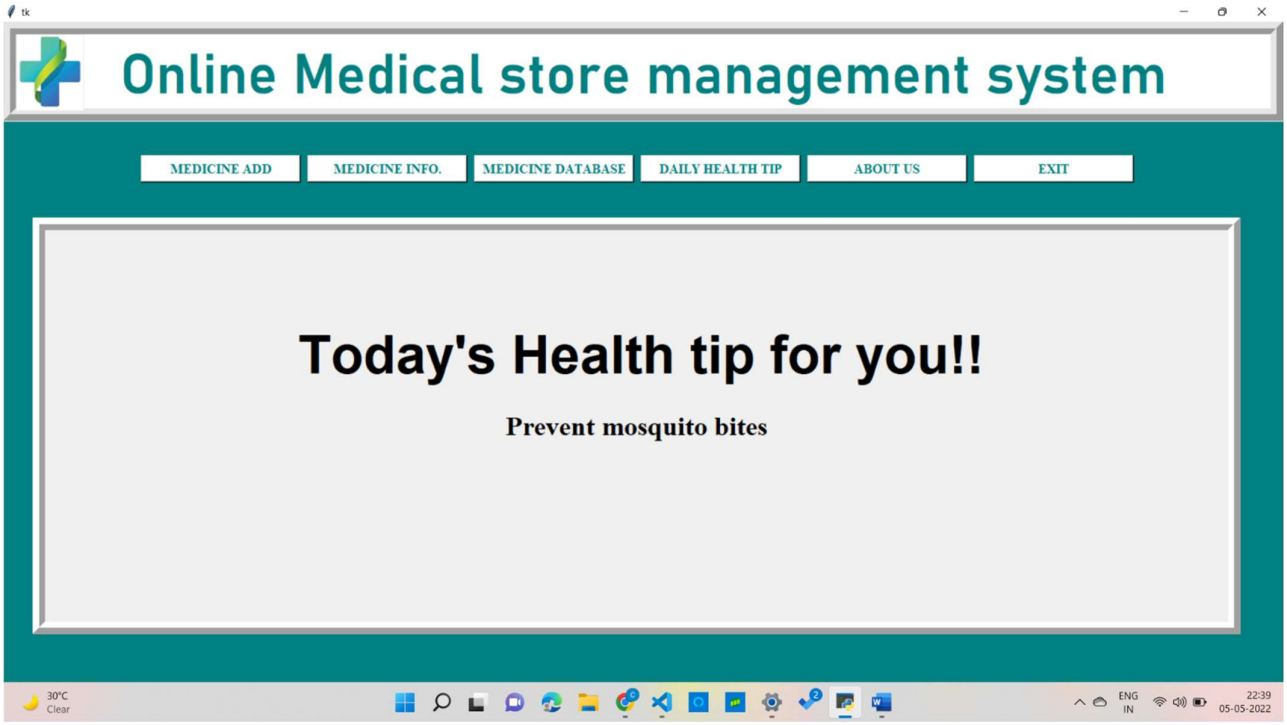
Eat a healthy diet



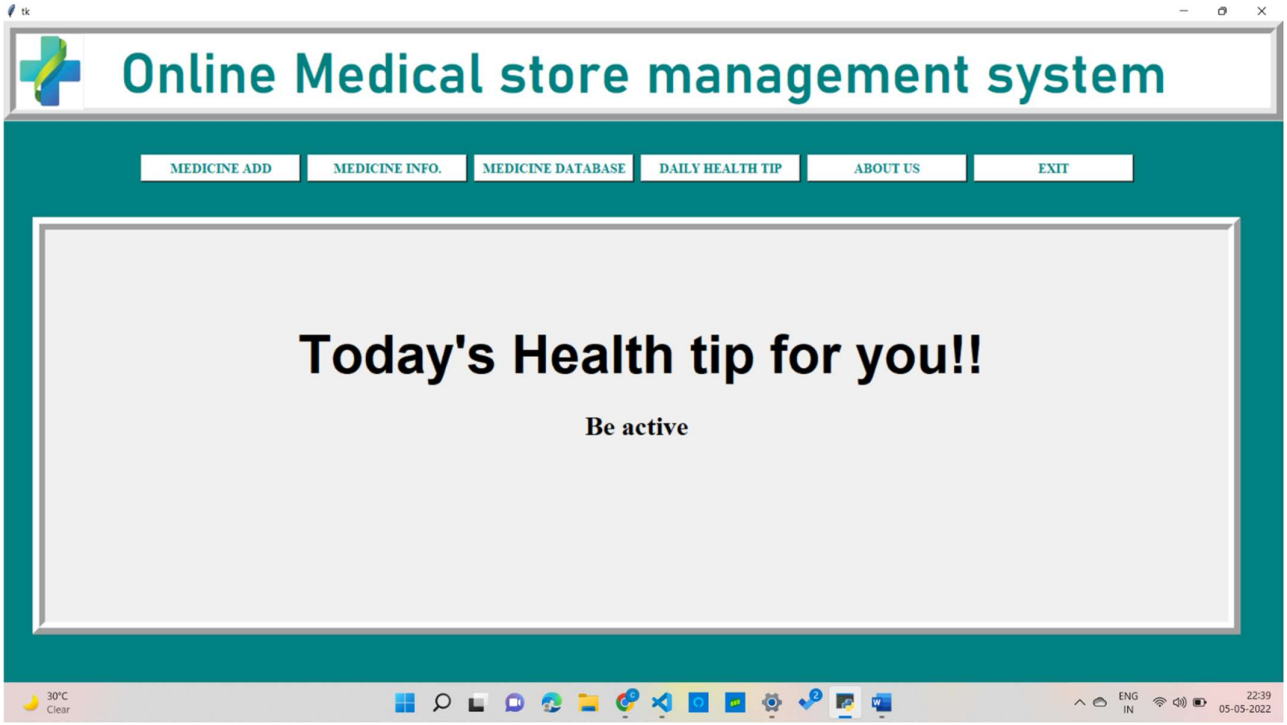
Have regular check-ups



Drink only safe water



Prevent mosquito bites



Be active

ABOUT US:

The next button is the about us button where you will get the basic knowledge about application. This contains the text as follows:

The text in about us is as follows

:

Our Online Medical store management system will manage record of all medicines, price details, availability of medicines, number of medicines sold, brand details, and other important details. It deals with the recording and processing data that medicine store owners can easily manage the operations. Prescriptions are proper and supplied in precise amounts using Pharmacy Management software. The control gadget employs strict measures to shield the users from intruders or outsiders. One of such measures is the incapacity of a potential person to check in at the platform without the permission of the

directors. To check in at the platform, all important info such as: name, address, tax files, charge methods, and applicable licenses are dispatched to the administrator for correct verification. When the authenticity of those files had been proven, the administrator then creates the account for the patron and forwards the info for login to the control gadget. This mission proposes one-of-a-kind architectures for database systems. The centralized, distributed, parallel, and designs are offered every with its strengths and weaknesses and the concrete motives for his or her selection. The mission recommends the use of centralized databases until there is a want for else. Also, on account that the medicine bought thru the platform are for pharmaceutical purposes, and as such, are ordered in bulk, if a small amount of medicine is ordered, the directors are notified, who then confirms the order from the drugstore to make sure that an outsider has not received their login info. The project proposes a schema for dividing the roles and responsibilities for operating and maintaining the database system among the staff concerned.
In this the various operations are used...!

Medicine Add: - In this function we can add the reference number and the medicine name and then the data entered gets saved in the database successfully as well as in this we can add, update or delete the data entered in it and by single click can clear the data entered showing up in the white frame window.

Medicine Info: - In this we can get by adding the reference number and company name of medicine we can get various things like its LOT number, expiry date, dosage of medicine, issue date, its side effects, uses, precautions and mainly the price of the medicine.

Medicine Database: - In this we have can search the medicine by adding just reference number, medicine name or LOT number and then after clicking on search button we can get the details of that medicine in a proper way.

Daily Health Tip: - In this it will provide every time a new health tip for

motivating us to stay healthy and there is a uniqueness in it that whenever we reload or restart that page it always shows us an anew
health tip without repetition.

Log Out/EXIT: - In this simply clicking on it will a new logout window confirmation window will get open and after clicking on it will directly logout you from the Tkinter window.

Our applications provide the following things:

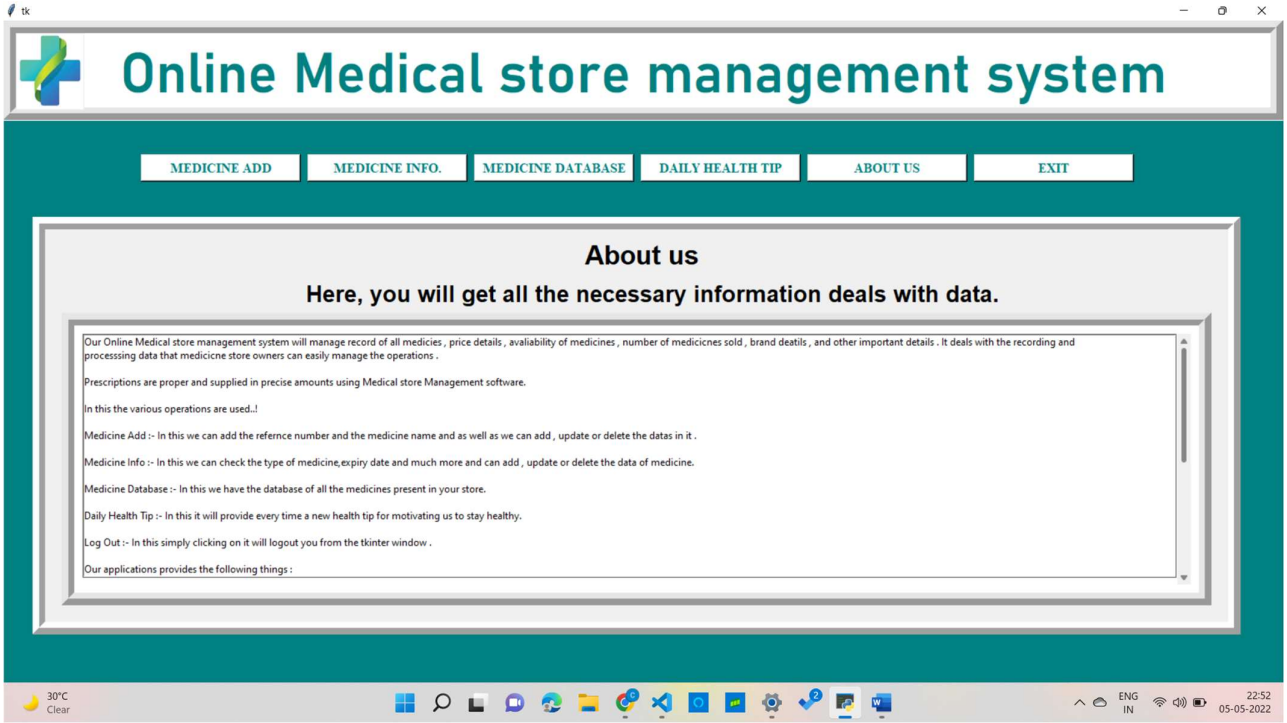
To provide a login facility to the customers.
To provide a list of all accessible medicines to the customers.
To update all the list of expired medicines pharmacist.
The main thing about system is to enhance and upgrade the existing system by increasing its efficiency and effectiveness. The software improves the working methods by replacing the existing manual system with the computer-based system.
In this, the transaction reports of the system can be retried as and when required. Thus, there is no delay in the availability of any information, whatever needed, can be captured very quickly and easily.

The key characteristics of Online Medical store Management System Project can be outlined as:

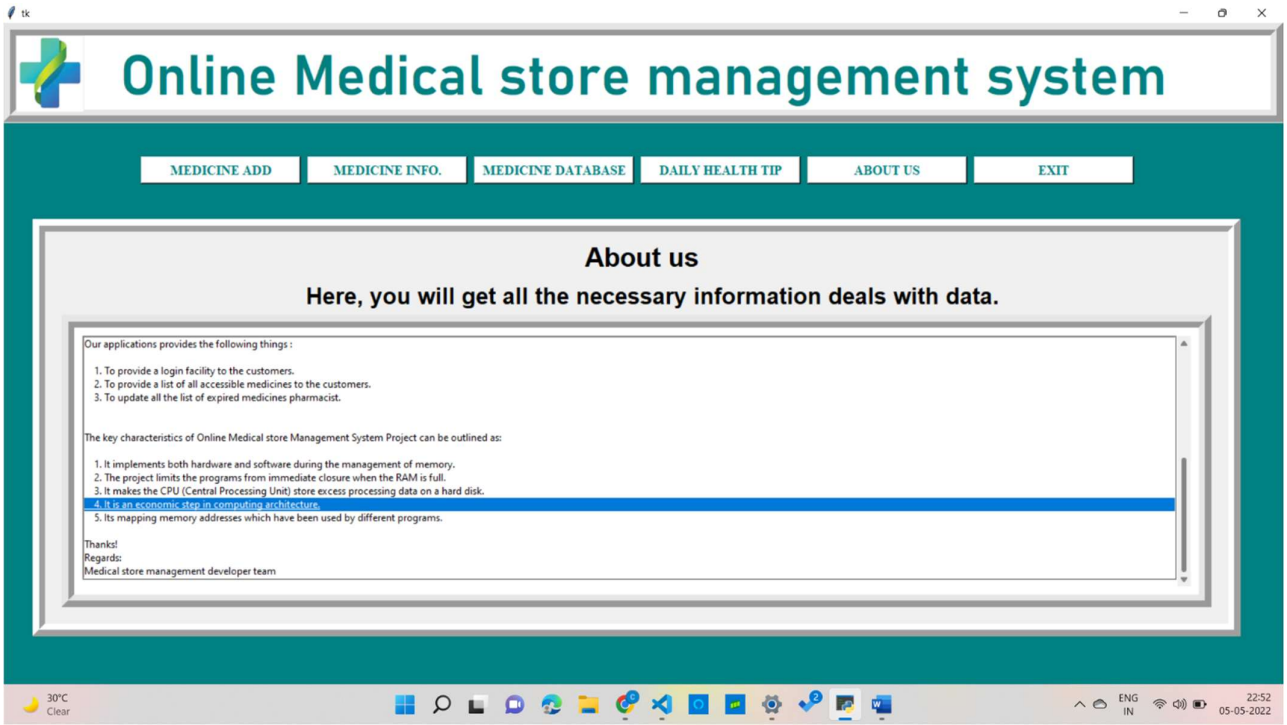
It implements both hardware and software during the management of memory.
The project limits the programs from immediate closure when the RAM is full.
It makes the CPU (Central Processing Unit) store excess processing data on a hard disk.
It is an economic step in computing architecture.
Its mapping memory addresses which have been used by different programs.

Here,

It is the view of the about us page from Online Medical store management system:



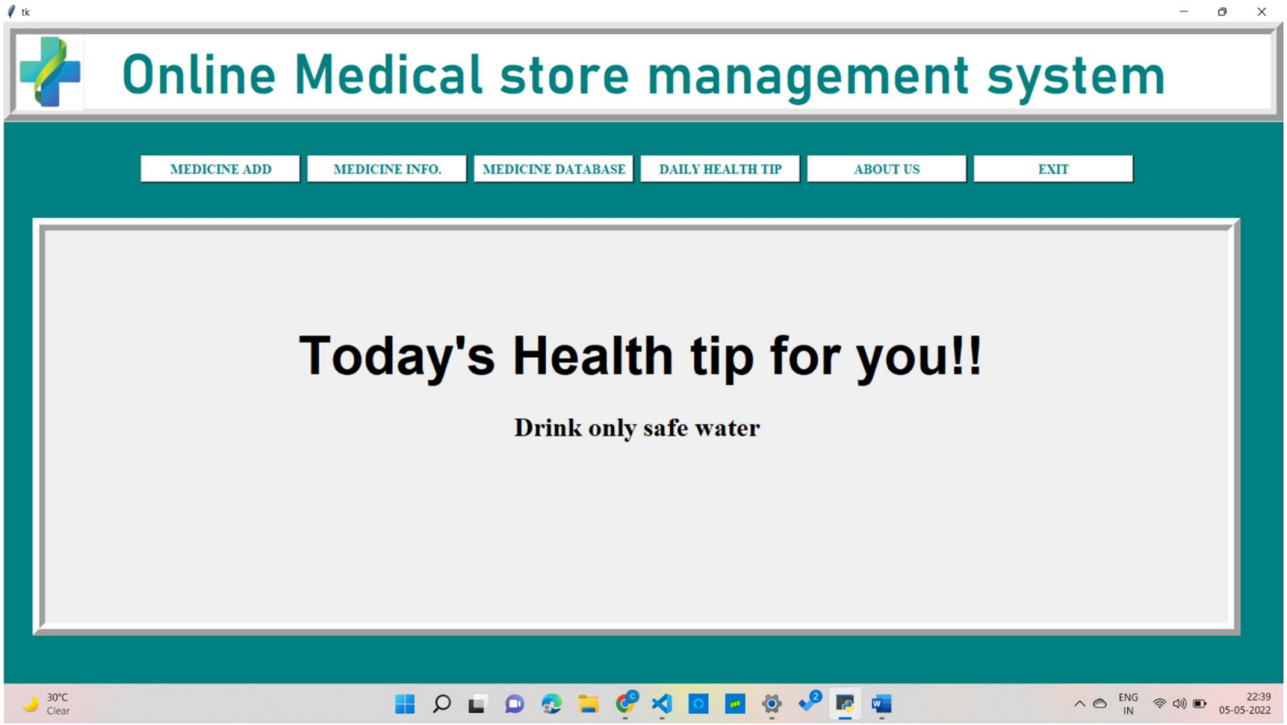
There is a scroll bar that is added here for easy navigation around the small guide book of the ONLINE MEDICAL STORE MANAGEMENT SYSTEM.



The main purpose to design this page is to provide an easy understanding and to have a look over the management system to get an idea about the use of the application. This includes description about every part of **Online medical store management system.**

At last, we are here with the **EXIT** button.

This button is intended to stop the functioning of the application and destroy the screen of the application.



Assume that you are just here while you are working on **online medical store management system.** And now you want to leave the system as you are done with your task. So, press **EXIT** button.

REFERENCES:

- Database Management Systems Book
  : - Johannes Gehrke and Raghu RamaKrishan.

- MySQL for Python Book
  : - Albert Lukaszewski .

- Learn Python
  : - Zed A. Shaw
  : - 3rd Edition

- Learn PostgreSQL
  : - Luca Ferrari, Enrico Pirozzi

- Python GUI Programming with Tkinter
  : - Alan D. Moore

- Web Programming
  : - W.B. Sabesta
  : - 3rd Edition

- History , 4th edition Trade paperback , Pharmaceutical press(2008)
- George A.Bender Great moments in Medicine , Cambridge university press(1967)
- Evangelos peroutsos: - Mastering Visual Basic 6.0
- Regulating Online Pharmacies: The Challenges Ahead.

- https://www.researchgate.net/publication/305426561_Online_Pharmaceutical_Management_System
- http://www.w3schools.com
- https://www.postgresql.org/
- https://wiki.python.org