**GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY**

**SECTOR – 16C, DWARKA, NEW DELHI-110078**

PROJECT REPORT ON

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF COMPUTER APPLICATION

SUBMITTED BY :- UNDER THE GUIDANCE OF

Bhanu Pratap Sharma (02512002019) Dr. Kusum Lata Bharti

Mohd Shuveb Saifi (00328502019) (Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE

COMM-IT CAREER ACADEMY

FC-31, SHEIKH SARAI, PHASE-II, NEW DELHI-110017

## CERTIFICATION

This is to certify that the project entitled “APOLLO PHARMACY” submitted to

“COMM-IT Career Academy”, affiliated to” Guru Gobind Singh Indraprastha University”, New Delhi-110075. Submitted for the partial fulfilment of the award of the “Bachelor of Computer Application (BCA)”, is an original work carried out by

“Bhanu Pratap Sharma (02512002019) and Mohd Shuveb Saifi (00328502019)”,

under the guidance of the “Dr. Kusum lata Bharti”.

This topic on which the report has been present is embodies by the student as an original contribution towards the discipline of computer science.

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Enrolment No.: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sign. Of the student: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Associate Professor

(Dr. Kusum Lata Bharti)

COMM-IT Career Academy

(Affiliated to GGSIP University)

**ACKNOWLEDGEMENT**

I owe many thanks to all the people who helped and supported me during the making of this project. I take this opportunity to express my profound to all sense of gratitude and respect to all those who helped me throughout the duration of this minor project.

I am thankful to “Dr. Kusum Lata Bharti” for guiding and correcting various documents of mine with attention and care.

Lastly, I would like to acknowledge and express my deep gratitude to all those who directly or indirectly assisted me in completing this major project report.

BHANU PRATAP SHARMA (02512002019)

Mohd Shuveb Saifi (00328502019)

BCA Vth Semester

COMM-IT CAREER ACADEMY

SHEIKH SARAI PHASE-II,

New Delhi-110017

## ABSTRACT

This project is insight into the design and implementation of a Pharmacy Handling System. The primary aim of is to improve accuracy and enhance safety and efficiency in the pharmaceutical store. Today management is one of the most essential features of all form. Management provides sophistication to perform any kind of task in a particular form. This is pharmacy handling system; it is used to manage most pharmacy related activities in the pharmacy.

# TABLE OF CONTENT

COVER PAGE

[CERTIFICATION I](#_Toc354972603)I

[ACKNOWLEDGMENT II](#_Toc354972604)I

[ABSTRACT IV](#_Toc354972605)

[TABLE OF CONTENT V](#_Toc354972606)

[TABLE OF FIGURES VI](#_Toc354972607)I

[LIST OF TABLES VI](#_Toc354972608)I

[**INTRODUCTION** 1](#_Toc354972609)

[1.0 INTRODUCTION 1](#_Toc354972610)

[1.1 BACKGROUND OF THE STUDY 1](#_Toc354972611)

[1.2 STATEMENT OF THE PROBLEM 2](#_Toc354972612)

[1.3 AIMS AND OBJECTIVES 3](#_Toc354972613)

[1.4 SCOPE AND LIMITATION 3](#_Toc354972614)

[**LITERATURE REVIEW** 5](#_Toc354972618)

[2.0 INTRODUCTION 5](#_Toc354972619)

[2.1 PHARMACY PRACTICE IN THE PAST 5](#_Toc354972620)

[2.2 THE PRESENT DAY PHARMACY 14](#_Toc354972621)

[2.3 THE FUTURE OF PHARMACY 16](#_Toc354972622)

[2.4 BENEFITS OF THE PROPOSED SYSTEM 19](#_Toc354972624)

[**SYSTEM ANALYSIS AND DESIGN** 21](#_Toc354972625)

[3.0 INTRODUCTION 21](#_Toc354972627)

[3.1 SYSTEM ANALYSIS 21](#_Toc354972628)

[3.1.1 ANALYSIS OF EXISTING SYSTEM 22](#_Toc354972629)

[3.2 REQUIREMENTS DEFINITION](#_Toc354972630) 23

[3.2.1 METHOD OF INFORMATION GATHERING 23](#_Toc354972631)

[3.3 SYSTEM DESIGN 23](#_Toc354972632)

[3.4 SYSTEM MODELLING 25](#_Toc354972633)

[3.5 SYSTEM FLOWCHART 25](#_Toc354972634)

[3.6 DFD(**Data Flow Diagram**) 27](#_Toc354972634)

[3.7 DATABASE DESIGN 30](#_Toc354972635)

[**SYSTEM IMPLEMENTATION** 33](#_Toc354972638)

[4.0 INTRODUCTION 33](#_Toc354972640)

[4.1 CHOICE OF PROGRAMMING LANGUAGE 33](#_Toc354972641)

4.2 SYSTEM TESTING AND DEBUGGING………………………………………......34

[4.3 SYSTEM DOCUMENTATION 35](#_Toc354972642)

[4.4 HARDWARE REQUIREMENT 35](#_Toc354972643)

4.5 SOFTWARE REQUIREMENT [35](#_Toc354972644)

[4.6 DATABASE SPECIFICATION 37](#_Toc354972644)

[4.7 MODULE DESCRIPTION 38](#_Toc354972645)

[4.8 SYSTEM MAINTENANCE 42](#_Toc354972646)

[**CONCLUSION AND RECOMMENDATION** 43](#_Toc354972647)

[5.0 SUMMARY 43](#_Toc354972649)

[5.1 CONCLUSION 43](#_Toc354972650)

[5.2 RECCOMMENDATION 44](#_Toc354972651)

APPENDIX…………………………………………………………………………...45

MAIN PAGE…………………………………………………………………...45

LOGIN PAGE……………………………………………………………….….…….47

DRUG REGISTRATION FORM AND

INFORMATION REGISTRATION FORM……………………………………….…52

DATABASE WINDOW…………………………………………...………………….80

# LIST OF FIGURES

Figure. 3.3.1 Waterfall Model of System Development Life Cycle……………….24

Figure 3.5 System flowchart…………………………………………………….26

Figure ER Diagram………………………………………………………….26

Figure 3.6 Login System Diagram……………………………………………....27

Figure 3.7.1 0-level DFD ………………………………………………….……....30

Figure 3.7.2 1-level DFD ………………………………………………….……....30

Figure 3.8 Database design……………………………………………………...31

Figure 4.1 Homepage…………………………………………………………...38

Figure 4.2 Admin Login Menu…………………………………………………39

Figure 4.3 Admin Main Menu………………………………………….….…...40

Figure 4.4 ADMIN MAIN MENU WITH ENTRIES………………………….40

Figure 4.5 ADMIN MAIN MENU WITH POPUP BUTTON MENU….….……….41

Figure 4.6 Database window……………………………………………………41

Figure 4.7 DATABASE WINDOW WITH NEW POPUP WINDOW……………….42

# LIST OF TABLES

Table 3.1 Pharmacy Table…………………………………………………. 31

Table 3.2 Pharma Table……………………………………………………. 32

# **INTRODUCTION**

# INTRODUCTION

Apollo Pharmacy is a pharmacy handling system that is designed to improve accuracy and to enhance safety and efficiency in the pharmaceutical store. It is a computer based system which helps the Pharmacist to improve inventory management, cost, medical safety etc.

The system allows the user to enter a manufacturing and expiry date for a particular product or drug during opening stock. The system will also give report showing the list of products expiry after a specified date before the product eventually expires. It also involves manual entry upon arrival of new batches of drugs and upon drug movement out of the pharmacy for a certain period, e.g. every month, the pharmacist may want to generate report for the movement of drugs in and out of the pharmacy, getting information about the drugs e.g. expiry date, date purchased, number of drug type left, location of a drug in the pharmacy.

At present, manual system is being utilized in the pharmacy. It requires the pharmacist to manually monitor each drug that is available in the pharmacy. This usually leads to mistakes as the workload of the pharmacist increases.

# 1.1 BACKGROUND OF THE STUDY

Due to the size and quality service of the pharmacy, the pharmacy has a very large customer base. These customers tend to visit the pharmacy for services mostly when they close from work. At this period, the number of customers that patronise the pharmacy is on the increase, thereby making the workload of the pharmacists much more tedious. This case makes it difficult for the pharmacist to attend to customers in a short period.

Meanwhile the pharmacist has to ensure satisfaction in services to keep their customers. The factors mentioned above, results in delay of the services being rendered to the customers, thereby slowing down sales and risk losing valuable customers in the long run.

# 1.2 STATEMENT OF THE PROBLEM

Pharmacy management has kept paper record in filing cabinets. Managing a very large pharmacy with records on papers will be tedious and difficult to keep track of inventories with regards to the drugs in the store, expiry date, quantity of drugs available based on the categories and their functions.

The pharmacist has to order drugs to replenish the already diminishing stock. In addition, ordering of drugs is being carried out manually. Significant amount of time is allocated for writing the order as the pharmacist needs to go through the stock balance and make rough estimate of the amount to order based on Figures.

Drugs are not supposed to be used after they have expired. This project work will prompt the pharmacist about drugs that are close to expiry, preventing those drugs from being sold and also providing solution to the earlier stated problems.

# 1.3 AIMS AND OBJECTIVES

The aim of this project is to develop a software for the effective management of a pharmaceutical store that will be able to achieve the following objectives:

* Ensuring effective policing by providing statistics of the drugs in stock.
* Maintaining correct database by providing an option to update the drugs in stock.
* Improving the efficiency of the system by ensuring effective monitoring of services and activities.
* To provide optimal drug inventory management by monitoring the drug movement in the pharmacy.
* To ensure that there exists a level of restricted access based on functionality and role.
* To ensure that the system is user friendly.

# 1.4 SCOPE AND LIMITATION

The scope of this project is limited to the activities of a pharmaceutical store which includes will improving health outcomes, reduce hospital and long term care admissions, enhance access and care in the Estate and surrounding communities and ensuring best use of resources, the use of a computer based management system for improving the efficiency of a pharmacy is needed and it is an essential part of any modern continuously evolving society.

The system will not be able to handle drug prescription, drug to drug interaction. The system will not be able to handle contraindication and polypharmacy in a prescription; this implies that these services will be manually completed by the pharmacist.

**LITERATURE REVIEW**

# INTRODUCTION

A pharmacist is a medical professional who dispenses drugs to patients according to a prescription ordered by a physician or other clinician. Pharmacists have an in-depth knowledge of the chemistry of various drugs and how they react in humans, and also how drugs interact with each other (Charles E. Rosenberg, 1980). Pharmacists must accurately measure and package medicine, ensuring its dosage and safety to be administered properly to a patient. While the pharmacist does not typically select or prescribe the medication, the pharmacist educates the patient on how to take the medication and what reactions or problems to be avoided.

As medication experts, pharmacists are concerned with safeguarding the public's health in matters relating to medication distribution and use and disease state management. Pharmacists play a vital role in improving patient care through the medicine and information they provide.

# 2.1 PHARMACY PRACTICE IN THE PAST

This gives an overview of events that have occurred in pharmacy practice through several ages both in the past and present, and also indicates possible events of the future.

2.1.1The History of Pharmacy

As long as there have been societies, there have been specialists and physicians whose sole purpose was to prepare and administer medicinal treatments. The earliest healers engaged in what is now called Pharmacognosy, which is the study and application of plants and herbs for healing. Evidence that early man used pharmacognosy to treat illness is indisputable, with archaeological discoveries attesting to the fact pre-dating even the development of farming or animal husbandry.

The beginnings of pharmacy are ancient. When the first person expressed juice from a succulent leaf to apply to a wound, this art was being practiced. In the Greek legend, Asclepius, the god of the healing art, delegated to Hygeia the duty of compounding his remedies. She was his apothecary or pharmacist. The physician-priests of Egypt were divided into two classes: those who visited the sick and those who remained in the temple and prepared remedies for the patients (Homan, 2008).

The pharmacy profession can be traced back at least as far as the Sumerian population, living in modern day Iraq from around 4000 BC, they used medicinal plants such as liquorice, mustard, myrrh, and opium. There were separate people who worked to prepare medicines, as a separate role from diagnosis and treatment which was carried out by medics. These precursors to pharmacists also combined their role with that of a priest. The Sumerians wrote the earliest surviving prescriptions from at least 2700 B.C. – so nearly 5000 years ago (Griggs, 1999).

The Ancient Egyptians had specific preparers of medicine, known as Pastophor. Pharmacy was viewed as a high status branch of medicine, and again, like the Sumerians, these pharmacists were also priests who worked and practised in the temples (Anderson, 2005).

From surviving papyrus scrolls, notably the Ebbers Papyrus which dates from 1500 BC, we know that the Egyptians made and used infusions, ointments, lozenges, suppositories, lotions, enemas, and pills. The Ebbers Papyrus includes 875 prescriptions and 700 drugs. Meanwhile, in China in about the same era (2000 BC), a man called Shen Nung wrote the first native herbal, which contained descriptions of 365 plant-based drugs (Anderson, 2005).

Stalls and shops selling medicinal goods existed around 1900 B.C. in the town of Sippara on the Euphrates River. However, the earliest recorded shop dealing with sales of medicines in London was opened in 1345.

In ancient Greece and Rome and during the middle Ages in Europe, the art of healing recognized a separation between the duties of the physician and those of the herbalist, who supplied the physician with the raw materials from which to make medicines. The Arabian influence in Europe during the 8th century had however brought about the practice of separate duties for the pharmacist and physician. The trend toward specialization was later reinforced by a law enacted by the city council of Bruges in 1683, forbidding physicians to prepare medications for their patients. In America, Benjamin Franklin took a pivotal step in keeping the two professions separate when he appointed an apothecary to the Pennsylvania Hospital.

The development of the pharmaceutical industry since World War II led to the discovery and use of new and effective drug substances. It also changed the role of the pharmacist. The scope for extemporaneous compounding of medicines was much diminished and with it the need for the manipulative skills that were previously applied by the pharmacist to the preparation of pills, plasters, and potions (Rosenberg, 2008).

The pharmacist continues, however, to fulfil the prescriber’s intentions by providing advice and information; by formulating, storing, and providing correct dosage forms; and by assuring the efficacy and quality of the dispensed or supplied medicinal product.

2.1.2 Origin and Development of Pharmacy

Before the dawn of history**,** ancient man learned from instinct, from observation of birds and beasts. Cool water, a leaf, dirt, or mud was his first soothing application. By trial, he learned which served him best. Eventually, he applied his knowledge for the benefit of others (Mathews, 1962).

**Pharmacy in ancient Babylonia**: Babylon, jewel of ancient Mesopotamia, often called the cradle of civilization. It provides the earliest known practice of the art of the apothecary. Practitioners of healing of this era (about 2600 B.C.) were priest, pharmacist and physician, all in one.

**Pharmacy in ancient China**: Chinese Pharmacy stems from Shen Nung (about 2000 B.C.), an emperor who sought out and investigated the medicinal value of several hundred herbs. Medicinal plants include podophyllum, rhubarb, ginseng, stramonium, cinnamon bark, ephedra etc.

**Days of the Papyrus Ebers:** "Papyrus Ebers" (1500 B.C.) is the best known and most important pharmaceutical record. It is a collection of 800 prescriptions mentioning 700 drugs.

**Theophrastus - Father of Botany:** Theophrastus (about 300 B.C.), was the greatest early Greek philosophers and natural scientists, is called the "father of botany." His observations and writings dealing with the medical qualities and peculiarities of herbs are unusually accurate, even in the light of present knowledge.

**Terra Sigillata - An Early "Trademarked" Drug:** Man learned early of the prestigious advantage of trademarks as a means of identification of source and of gaining customers' confidence. One of the first therapeutic agents to bear such a mark was Terra Sigillata (Sealed Earth), a clay tablet originating on the Mediterranean island of Lemnos before 500 B.C. (George A. Bender, 1967)

**Dioscorides - A Scientist Looks At Drugs:** Pedanios Dioscorides (first century A.D.), contributed mightily to such a transition in Pharmacy. He observed, recorded and spread the excellent rules for collection of drugs, their storage and use. His texts were considered basic science as late as the sixteenth century.

**Galen - Experimenter in Drug Compounding:** He practiced and taught both Pharmacy and Medicine in Rome. His principles of preparing and compounding medicines ruled in the Western world for 1,500 years; and his name still is associated with that class of pharmaceuticals compounded by mechanical means - galenicals. He was the originator of the formula for a cold cream.

**Damian and Cosmas - Pharmacy's Patron Saints:** twin ship of the health professions, Pharmacy and Medicine, is portrayed by Damian, the apothecary, and Cosmas, the physician.

**Monastic Pharmacy:** During the middle ages remnants of the western knowledge of pharmacy and medicine were preserved in the monasteries (fifth to twelfth centuries). Manuscripts from many islands were translated or copied for monastery libraries. The monks gathered herbs and raised them in their own herb gardens. These they prepared according to the art of the apothecary for the benefit of the sick and injured. Gardens such as these still may be found in monasteries in many countries.

**The First Apothecary Shops:** The Arabs separated the arts of apothecary and physician, establishing in Bagdad late in the eighth century the first privately owned drug stores. They preserved much of the Greco-Roman wisdom developing with the aid of their natural resources syrups, confections, conserves, distilled waters and alcoholic liquids.

**Avicenna - The "Persian Galen":** Among the brilliant contributors to the sciences of Pharmacy and Medicine during the Arabian era was one genius who seems to stand for his time - the Persian, Ibn Sina (about 980-1037 A.D.), called Avicenna by the Western world. He was a Pharmacist, poet, physician, philosopher and diplomat. He gave contribution to the sciences of pharmacy and medicine by his pharmaceutical teachings.

**Separation of Pharmacy And Medicine:** In European countries, public pharmacies began to appear in the 17th century. In Sicily and southern Italy, pharmacy was separated from Medicine.

**The First Official Pharmacopoeia:** The idea of a pharmacopoeia with official status, to be followed by all apothecaries, originated in Florence. The Nuovo Receptario, originally written in Italian, was published and became the legal standard for the city-state in 1498.

**The Society of Apothecaries of London:** In 1617, Francis Bacon (philosopher-politician) formed a separate company known as the "Master, Wardens and Society of the Art and Mystery of the Apothecaries of the City of London". This was the first organization of pharmacists in the Anglo-Saxon world (Mathews, 1962).

**Scheele - Greatest of the Pharmacists-Chemists:** During his few short years, Carl Wilhelm Scheele gave to the world discoveries that have brought its people incalculable advantages. He made thousands of experiments, discovered oxygen, chlorine, prussic acid, tartaric acid, tungsten, molybdenum, glycerine, nitro-glycerine, and countless other organic compounds that enter into today's daily life, industry, health, and comfort.

**The American Pharmaceutical Association:** Under leadership of its first President, Daniel B. Smith, and first Secretary, William Procter, Jr., the twenty delegates launched The American Pharmaceutical Association and opened membership to "All pharmacists and druggists" of good character who subscribed to its Constitution and to its Code of Ethics. The Association continues to serve Pharmacy today (Bender, 1967).

**European and American Pharmacy Meet:** European and American Pharmacy groups met for the first time, at the Second International Congress of Pharmacy in Paris, France in 1867.

2.1.3 Some key dates in Pharmacy history

1820 The alkaloid quinine was first extracted from the bark of cinchona trees by two French chemists, Pierre Joseph Pelletier and Joseph Biename Caventou.

1874   Diamorphine or Heroin was first synthesised from morphine.

1883 First edition of The Extra Pharmacopoeia published, edited by William Martindale and Dr Wynn Westcott.

1899   Aspirin, was launched by the German company.

1910 Salvarsan, the first 'magic bullet' drug, effective against syphilis was discovered by Paul Ehrlich and Dr Sahachiro Hata.

1915   Medicine stamp duty was doubled as a wartime fundraiser.

1917 The Venereal Disease Act prohibited the advertising of medicines for VD and selling mixtures containing scheduled substances.  It introduced the concept of 'prescription only' medicines.

1922 The Dangerous Drugs Act regulated the import and sale of potential drugs of addiction including the derivatives of opium, cocaine and cannabis so widely used in proprietary remedies.

1928   Penicillin discovered by Alexander Fleming.

1938 The Food and Drugs Act prohibited the adulteration and mislabelling of drugs.

1939 The Cancer Act restricted the advertisement of products claiming to treat cancer.

1940 Under the Finance (No. 2) Act purchase tax was imposed on a range of goods including most drugs and medicines.

1941 The Pharmacy and Medicines Act repealed the old medicine stamp duty. It forbade the general advertisement of products claiming to treat a number of specific illnesses including Bright's disease, cataract epilepsy and TB, or to be effective in procuring an abortion.  For the first time manufacturers were required to list the active ingredients of products on their packaging.

1948 The National Health Service made prescription medicine available to all. Until the introduction, in the 1950s, and subsequent hefty increasing of prescription charges, proprietary medicines were no longer seen as a cheap alternative to seeing the doctor.

1961 Ibuprofen was first synthesised by a team at the Boots Pure Drug Company in December.

1964 Introduction of Adverse Drug Reaction 'yellow card' scheme in response to the thalidomide tragedy of 1961.

# 2.2 THE PRESENT DAY PHARMACY

The modern drugstore varies significantly from its ancient counterparts. While the proprietors of pharmacies in the far distant past were often making numerous medical decisions - diagnosing and treating patients without the consultation of physicians - pharmacists in the modern drugstore are tasked instead with the responsibility of evaluating the appropriateness and managing the dispensation of pharmaceuticals prescribed to patients under a doctor's care. Among the most important of pharmacist jobs is assuring that the patient has not been prescribed two drugs which will have an adverse interaction. An in-depth study of pharmacology is required to make such evaluations and in all states such positions are highly regulated and require testing before the issuance of a pharmacist's license.

Far from being just a clerk behind a counter, pharmacists also play a role in disease management and evaluation of test results. In hospitals, pharmacists are often in the role of interpreting chemical signatures in complex test results and advising physicians on treatment options as well as making doctors aware of new and possibly more effective drugs.

Many people interested in a career as a pharmacist have found that in recent times it is a quite reliable profession in times of economic hardship. Earnings for professional pharmacists continue to rise while the industry for new and returning workers is quite robust.

2.2.1 How to Become a Pharmacist - Education and Training

The history of pharmaceutical education has closely followed that of medical education. As the training of the physician underwent changes from the apprenticeship system to formal educational courses, so did the training of the pharmacist. The first college of pharmacy was founded in the United States in 1821 and is now known as the Philadelphia College of Pharmacy and Science. Other institutes and colleges were established soon after in the United States, Great Britain, and continental Europe. Colleges of pharmacy as independent organizations or as schools of universities now operate in most developed countries of the world.

The course of instruction leading to a bachelor of science in pharmacy extends at least five years. The first and frequently the second year of training, embracing general education subjects, are often provided by a school of arts and sciences. Many institutions also offer graduate courses in pharmacy and cognate sciences leading to the degrees of Master of Science and doctor of philosophy in pharmacy, pharmacology, or related disciplines. These advanced courses are intended especially for those who are preparing for careers in research, manufacturing, or teaching in the field of pharmacy.

Since the treatment of the sick with drugs encompasses a wide field of knowledge in the biological and physical sciences, an understanding of these sciences is necessary for adequate pharmaceutical training. The basic five-year curriculum in the colleges of pharmacy of the United States, for example, embraces physics, chemistry, biology, bacteriology, physiology, pharmacology, and many other specialized courses. As the pharmacist is engaged in a business as well as a profession, special training is provided in merchandising, accounting, computer techniques, and pharmaceutical jurisprudence.

Pharmacists graduating from college today are required to have a PharmD, or doctorate of Pharmacy degree. College students can start a four-year pharmacy program after successfully completing two years of undergraduate coursework and earning a passing score on the PCAT (Pharmacy college admission test). Coursework in pharmacy and pre-pharmacy includes chemistry, physics, biology, anatomy, and physiology.

Additionally, PharmD students must complete a series of rotations in a variety of clinical and pharmaceutical settings. The length and quantity of rotations varies, but the average PharmD program requires 7-10 rotations, each of which is 4-6 weeks in length.

If a student knows early in his or her college career that they would like to become a pharmacist, one could graduate with a PharmD in about 6 years. Many college students do not decide until later in college or after college to become a pharmacist; therefore, many pharmacists have completed eight years of college.

## 2.3 THE FUTURE OF PHARMACY

Automation will assist with this, because anything that streamlines and automates the dispensing and distribution process will obviously free the pharmacist to fulfil more of a clinical role as well," said James Owen, BS Pharm, PharmD, director of professional practice for the American Pharmacists Association (APhA).

Simply put, pharmacists and pharmacies of tomorrow must find ways to increase profit margins by reducing the cost of prescribing, embracing technology, and focusing more on patient counselling, immunizations, education, and other natural offshoots of their clinical expertise. Some experts predict that the future of pharmacy will embrace the clinical skills and care pharmacists have always been qualified to deliver.

Technological options available to pharmacies today are as numerous as they are convenient. As the healthcare system becomes more digital, the most exciting thing is that pharmacists will have the potential to become more connected to patients and have more time for patient-centred activities."

## 2.3.1 Categories of Some of the technologies in the future are:

* Resources for clinical decision support.
* Error prevention and quality assurance.
* Advances in bar-coding and even radio-frequency identification technology will help pharmacists verify the appropriate use of medications.
* Telecommunications infrastructure. For example, to be able to send a patient a text message reminder to take meds or measure blood glucose levels. This will lead to better self-care management behaviours.
* Automation and robotics.
* Collaborative practice connectivity, achieved through a combination of electronic prescribing (e-prescribing) and the ability to connect to and exchange data with labs and physician offices.
* Delivery service support, using GPS-driven telecommunications systems for routing, tracking, order status, dispatch, locating, and oversight.
* Pharmacy surveillance and security systems, which will create a greater level of security for pharmacies even when pharmacists are off-site.

Pharmacies that do not step up to this challenge will lose the ability to handle important drugs, and this will mean loss of some of their most valuable prescriptions and patients.

Several companies sell software and hardware that increase the efficiency of pharmacies by managing workflow.

The pharmacist will play a significant role in making contributions to patient care, and these contributions will be documented using technology and transmitted using technology, and that care will be recorded in the EHR (electronic health records) in the future. Pharmacists will be integral team members as far as the care of patients is concerned."

Improved care and reduced errors will be the cornerstones of this relationship between pharmacists and automation.

2.3.2 The physical layout or appearance of tomorrow's pharmacies

Some experts say it depends on the individual pharmacy and the type of automation being implemented.

According to Bill G. Felkey, BA, MS, professor of healthcare informatics in the department of pharmacy care system at Auburn University in Alabama predicted that many pharmacies across the country will choose to move the pharmacist to a more conspicuous forward location in their stores. With the pharmacist in front, however, you can put the technology behind closed doors or you can believe that patients will be fascinated to see how these robotics work. There will exist a mixed reception, depending upon where the pharmacy is and upon how well the patients receive the display of all these technologies.

Muller said that dispensing machines will have a definite impact on layout. It has to be easily accessible to staff, but not located in an area that causes traffic and bottlenecks. Workflow also affects the logical placement of a dispensing machine. If the machine counts but does not label and vial, it needs to be located in the technician area and positioned so all techs have easy access. If the machine labels and vials, and only requires the final pharmacist check, it needs to be located between the pharmacists and technicians, depending on who is putting the prescription in the bag. Automation needs to fit into workflow, not just be a part of it.

Christopher Thomsen, vice president, business development, Kirby Lester Incorporation agreed, noting that even in a situation where a chain may have gone to a central-fill system, where, say 20 to 30 per cent of the volume is moved off-site, it still needs to determine how to best use automation to address the remaining on-site dispensing requirements.

Which is why, said Muller, you would not just add a machine into an existing layout; some thought and care must be given to properly reflow the behind-the-counter space to get the most from your investment.

The bottom line is that everything needs to move toward a more efficient flow.

# 2.4 BENEFITS OF THE PROPOSED SYSTEM

The proposed system would be designed to help make the rigorous activities carried out in a pharmacy much easier by providing the statistics of drugs in stock, monitoring drug movement in the pharmacy and ensuring effective policing of the activities in the pharmacy.

The new system will be designed to provide the following benefits in the interest of the pharmacy;

* The system would enhance management services and improve productivity.
* The system would enhance User/System interface.
* The system would be cost effective.
* The system would improve information quality and accessibility.

# **SYSTEM ANALYSIS AND DESIGN**

# 3.0 INTRODUCTION

System is a collection of an interrelated components that works together to achieve a purpose. System analysis is referred to the systematic examination or detailed study of a system in order to identify problems of the system, and using the information gathered in the analysis stage to recommend improvements or solution to the system.

System design is an abstract representation of a system component and their relationship and which describe the aggregated functionality and performance of the system. System design is also the overall plan or blueprint for how to obtain answer to the question being asked. The design specifies which of the various type of approach

## 3.1 SYSTEM ANALYSIS

System analysis is the study of sets of interacting entities, including computer systems analysis. This field is closely related to requirements analysis or operations research. It is also "an explicit formal inquiry carried out to help someone identify a better course of action and make a better decision than he might otherwise have made. System Analysis is a methodology that involves the application of systematic approaches to collects facts about an existing system with the aim of improving it or replacing it with more efficient system within the context of the available resources. In other words, System analysis can also be viewed as the process of investigating a system, identifying problems and using the information to recommend improvements to the system.

## 3.1.1 ANALYSIS OF EXISTING SYSTEM

Before we analyse the design of the proposed system, we need to carefully highlight the problems of the existing system so as to avoid recurrence. This analysis serves as a pointer on how to embark on building the proposed system that will help the Pharmacist provide optimal drug inventory management by monitoring the drug movement and state in the pharmacy. The problems of the current system should be outlined. Below are some of the problems associated with the existing system;

* Significant amount of time is allocated for writing the order as the pharmacist needs to go through the stock balance and make rough estimate for the amount to order based on Figures.
* The state of drugs in stock is manually checked.
* Mistake of selling expired drugs to customers.
* Too much workload on employees
* Filing cabinet in the pharmacy with paper record.

3.1.2 ANALYSIS OF PROPOSED SYSTEM

From the problems listed in the existing system, the implementation of the proposed system shall focus on;

* Pharmacists having access to the proposed system at any time.
* Ensuring effective policing by providing statistics of the drugs in stock.
* Improving the efficiency of the system by ensuring effective monitoring of services and activities.
* Generating report within a specified period of time.
* Reducing the employees’ workload.

3.2 REQUIREMENTS DEFINITION

Preliminary investigation plays an important role in developing a satisfactory requirement. Its’ as a result of thorough investigation of how the current or the existing system works using the facts gathered at the preliminary investigation that leads to focusing on the possibility of replacing the existing system or improving upon the existing system. This task involves information gathering.

## 3.2.1 METHOD OF INFORMATION GATHERING

Collection of fact is the act of getting and gathering information from various sources in order to be able to compose the project. Data used for designing of the system were gathered through several means. Therefore, the method used in the design and collections of information from various sources are as follows:

* Collecting and analysing existing materials on the project topic, written by different expert.
* Studying the present system in detail and the organizational style.
* Knowing and understanding the input and output processes of the existing system.
* Interviews: A qualitative form of interview was conducted in the pharmacy to know the equipment needed, and the mode of operation of the old system.
* Primary data: This source has to do with the text book contacted for the development of this project.

## 3.3 SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements through system modeling. One could see it as the application of [systems theory](http://en.wikipedia.org/wiki/Systems_theory) to [produce development](http://en.wikipedia.org/wiki/Product_development). The design of this system will be user friendly. It shall be designed in such a way that employees will be able to navigate easily through the information supplied on the system.

In other words, system design consists of design activities that produce system specifications satisfying the functional requirements that were developed in the system analysis process. System design specifies how the system will accomplish. System design is the structural implementation of the system analysis.

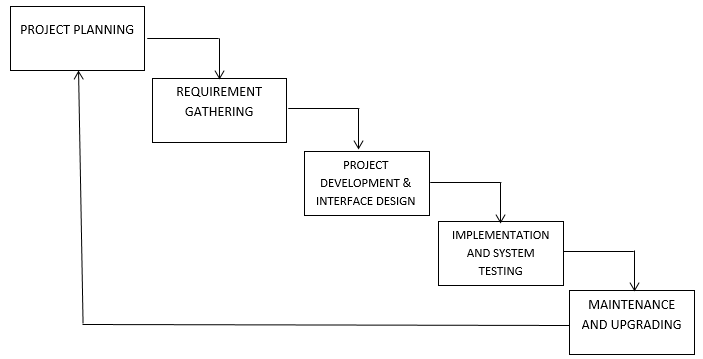


FIG. 3.3.1 WATERFALL MODEL OF SYSTEM DEVELOPMENT LIFE CYCLE

The diagram above is a system development life cycle that illustrates how the design of the project is broken down into five different phases, which are Project Planning, Requirement Gathering, Project Implementation and Interface Design, Implementation and System Testing, Maintenance and System Upgrading.

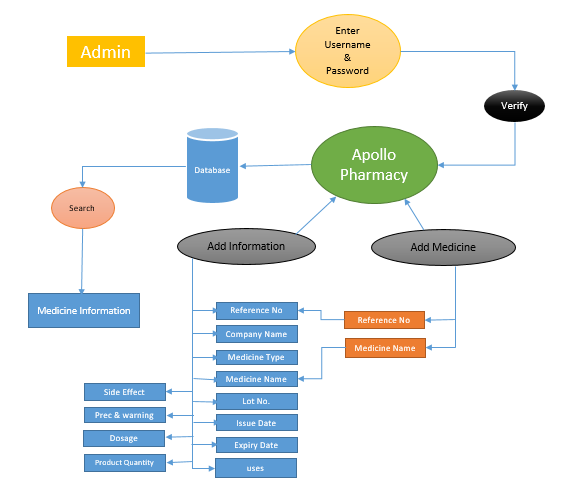
The proposed Apollo Pharmacy for Pharmacy company and Stores will start with project planning by determining the users of the system, aims and objectives of the project. After these, extensive research will be done to determine how to design an effective system, as well as to review the current system. Then, the design was with an initial prototype of the system, and then refined it based on their suggestions. Phases of analysis, design and implementation were performed iteratively until users and designers agreed on a final system specification. At this point, the project could move to the final implementation phase.

## 3.4 SYSTEM MODELLING

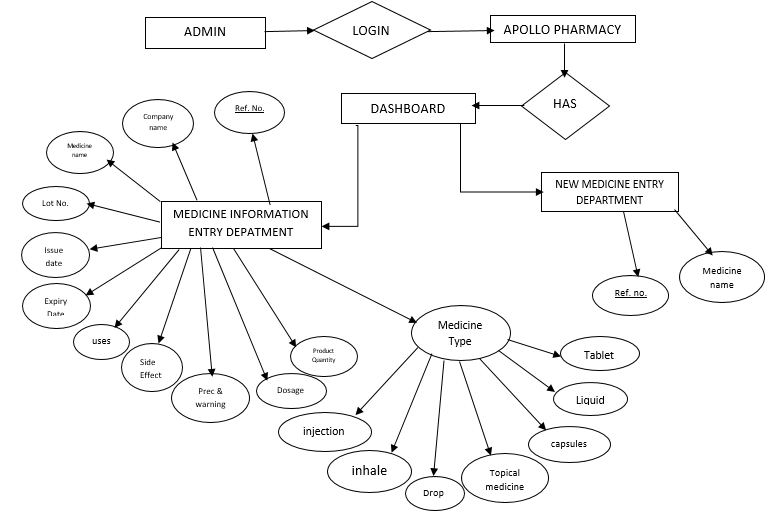
During the system requirements and design activity, systems may be modelled as a set of components and relationships between these components. These are normally illustrated graphically in a system architecture model that gives the reader an overview of the system organisation. System modelling helps to give more detailed system specifications which are in form of graphical representations that can describe problem to be solved or the system that is to be developed. Because of the graphical representations used, models are often more understandable than detailed natural language description of the system requirements. Examples of such modelling tool is a System Flowchart.

## 3.5 SYSTEM FLOWCHART

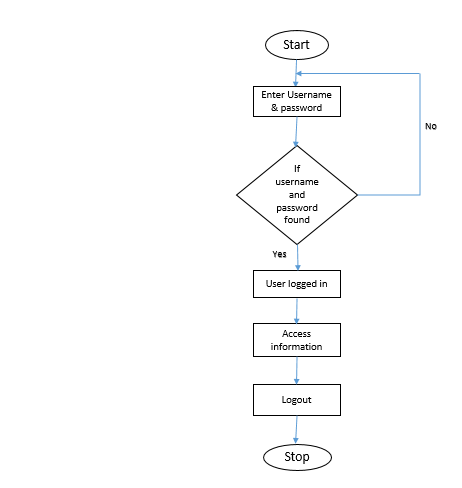
System flowchart is a type of [diagram](http://en.wikipedia.org/wiki/Diagram) that represents an [algorithm](http://en.wikipedia.org/wiki/Algorithm) or [process](http://en.wikipedia.org/wiki/Process_%28science%29), showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic [representation](http://en.wikipedia.org/wiki/Knowledge_representation_and_reasoning) can give a step-by-step solution to a given [problem](http://en.wikipedia.org/wiki/Problem_solving). Process operations are represented in these boxes, and arrows connecting them represent flow of control. Flowcharts are used in analysing, designing, documenting or managing a process or program in various fields. Different symbols are used in the flowchart to represent input, output, decision, connectors and process.



ER Diagram



# **3.6 Login System diagram**



# **3.7 DFD (Data Flow Diagram)**

**DFD** is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

#### **Components of DFD**

The Data Flow Diagram has 4 components:

* **Process**  
  Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence
* **Data Flow**  
  Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.
* **Warehouse**  
  The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updation.
* **Terminator**  
  The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

#### **Rules for creating DFD**

* The name of the entity should be easy and understandable without any extra assistance (like comments).
* The processes should be numbered or put in ordered list to be referred easily.
* The DFD should maintain consistency across all the DFD levels.
* A single DFD can have maximum processes upto 9 and minimum 3 processes.

#### **Levels of DFD**

DFD uses hierarchy to maintain transparency thus multilevel DFD’s can be created. Levels of DFD are as follows:

* 0-level DFD
* 1-level DFD:
* 2-level DFD:

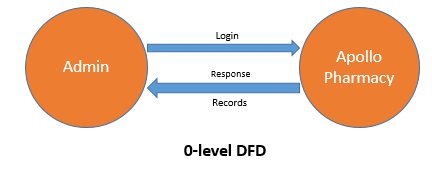
#### **Advantages of DFD**

* It helps us to understand the functioning and the limits of a system.
* It is a graphical representation which is very easy to understand as it helps visualize contents.
* Data Flow Diagram represent detailed and well explained diagram of system components.
* It is used as the part of system documentation file.
* Data Flow Diagrams can be understood by both technical or nontechnical person because they are very easy to understand.

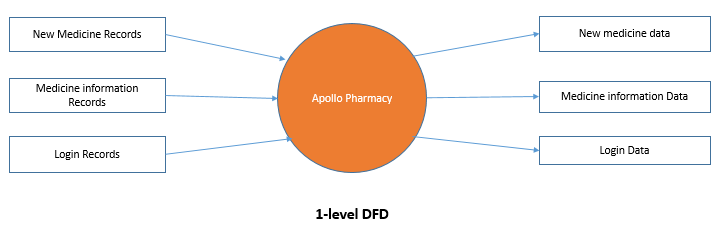
#### **Disadvantages of DFD**

* At times DFD can confuse the programmers regarding the system.
* Data Flow Diagram takes long time to be generated, and many times due to this reasons analysts are denied permission to work on it.

**FIG 3.7.1 0-level DFD**

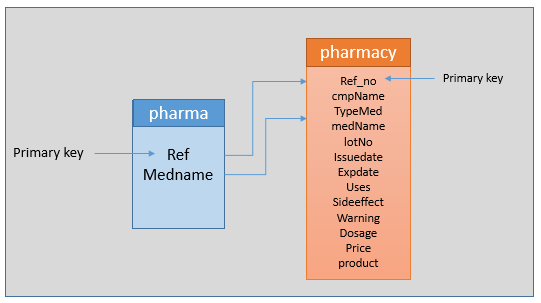


**FIG 3.7.2 1-level DFD**



## 3.8 DATABASE DESIGN

This is a shared collection of data that are related or files that are to meet the immediate need of authorized users. These data **may** be in form of text, numeric, date or encoded images.



PHARMA TABLE

| **Field Name** | **Field Type** | **Field Length** | **Description** |
| --- | --- | --- | --- |
| Ref | int | pk | Reference no |
| MedName | Text | (20) | Medicine Name |

Table 3.0 above serves as a repository for pharmacy details, that will be able to have access to the entire details on the application. It is a table that stores the add new medicine details.

MEDICINE INFORMATION TABLE

# **Table 3.1 Drug information table: This table is named pharmacy; it depicts the information of the drugs in the pharmacy.**

Table 3.1 shows the information about the drugs in the pharmacy and is being queried from the database on the drug registration page to show all the drugs for user to select.

| **Field Name** | **Field Type** | **Field Length** | **Description** |
| --- | --- | --- | --- |
| Ref\_no | Int | PK | **reference No** |
| cmpName | Varchar | (20) | **Company Name** |
| TypeMed | Varchar | ( 20) | **Type of Medicine** |
| medName | Varchar | ( 20) | **Tablet Name** |
| lotNo | Varchar | (20 ) | **Lot No** |
| Issuedate | Varchar | ( 20) | **Issue Date** |
| Expdate | Varchar | (20 ) | **Exp Date** |
| uses | Varchar | ( 20) | **Uses** |
| sideeffect | Varchar | (20) | **Side effect** |
| warning | Varchar | (20) | **Warning** |
| dosage | Varchar | ( 20) | **Dosage** |
| Price | Varchar | ( 20) | **Price** |
| product | Varchar | (20 ) | **Product Quantity** |

# **SYSTEM IMPLEMENTATION**

# 4.0 INTRODUCTION

System implementation is a stage in system life cycle whereby a new system is developed, installed and made ready for use. It is this stage that all details and key point in the requirement specification are practicalised. System implementation therefore, is a very essential stage in which its success determines to a great extent the success of the new system. At this instance, after all is said and done the system is duly ready to be implemented (Apollo Pharmacy).

System design is concerned mainly with the coordination of activities, job procedures and equipment utilization in order to achieve organizational objectives. It addresses data input and output data, processing and interface.

## 4.1 CHOICE OF PROGRAMMING LANGUAGE

Choosing a programming language depends on your language experience and the scope of the application you are building. While small applications are often created using only one language, it is not uncommon to develop large applications using multiple languages.

The propose application to be built is not a web based application that needs internet facilities to function but a standalone application.

The choice of programming language to use for this programme is Python. The structure of the Basic programming language is very simple, particularly as to the executable code.

Python has many new and improved features such as inheritance, interfaces, and overloading that make it a powerful object-oriented programming language. It is particularly easy to develop [graphical user interfaces](http://en.wikipedia.org/wiki/Graphical_user_interface) and to connect them to handler functions provided by the application.

Python fully integrates the Tkinter and the common language runtime, which together provide language interoperability, garbage collection, enhanced security, and improved versioning support.

4.2 SYSTEM TESTING AND DEBUGGING

Testing is an integral part of software development processes. This is to ensure that the quality requirement of the application is not compromised by testing and debugging program modules before they are integrated, testing the system to ensure an effective inter-operability after integration.

Debugging has to do with fixing of errors encountered during program execution. System testing deals with the real life testing of the system, to ascertain how far it has gone in carrying out the expected task. This was carried out in two phases.

Number one is the source code testing which examine the logic of the program. Secondly, the specification testing which involves the examination of the system as regard to what it should do and how it should be done given specific conditions. This includes inputting data, collecting its output and comparing it with the output of the old system and assessing it to see if it can replace the old system.

# 4.3 SYSTEM DOCUMENTATION

System documentation is a crucial aspect of implementation process. It describes the working of components and serves as a method of communication between application developers and users. It also helps future analysis of application either by the same or different system analysts and developers.

# 4.4 HARDWARE REQUIREMENT

* A minimum hard disk space of 20 Gigabytes (GB)
* RAM size of 2GB
* Pentium 4 dual processor CPU
* A VGA colour monitor
* Mouse
* Keyboard

4.5 SOFTWARE REQUIREMENT

* Python
* Tkinter (GUI)
* Mysqlite3
* **PYTHON: -**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

* **Tkinter**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.  
**To create a tkinter app:**

* Importing the module – tkinter
* Create the main window (container)
* Add any number of widgets to the main window
* Apply the event Trigger on the widgets.
* **SQLite3**

**Python SQLite3** module is used to integrate the SQLite database with Python. It is a standardized Python DBI API 2.0 and provides a straightforward and simple-to-use interface for interacting with SQLite databases. There is no need to install this module separately as it comes along with Python after the 2.5x version.

# 4.6 DATABASE SPECIFICATION

A database is a single file which consists of structured data and records which are stored in minimum or no duplication of data. It is therefore a constructed, consistent and controlled pool of data. A good database must be common to all users and independent of the programs which use it to generate output.

However, SQLITE3 was used as the database application tool for designing the database management system. The database management system is limited only to database administrator (Management). Whilst the system designer / developer / programmer is responsible for maintaining and upgrading of the database and the whole software.

# 

# **4.7 MODULE DESCRIPTION**

HOME PAGE



Figure 4.1 showing HOME PAGE

Figure 4.1 shows the select login page. It is the first interface that appears on the screen when the application is being loaded. This interface displays the name of the application and some other information about the software. The page consists of logins that exist for several other levels in the application.

ADMIN LOGIN MENU

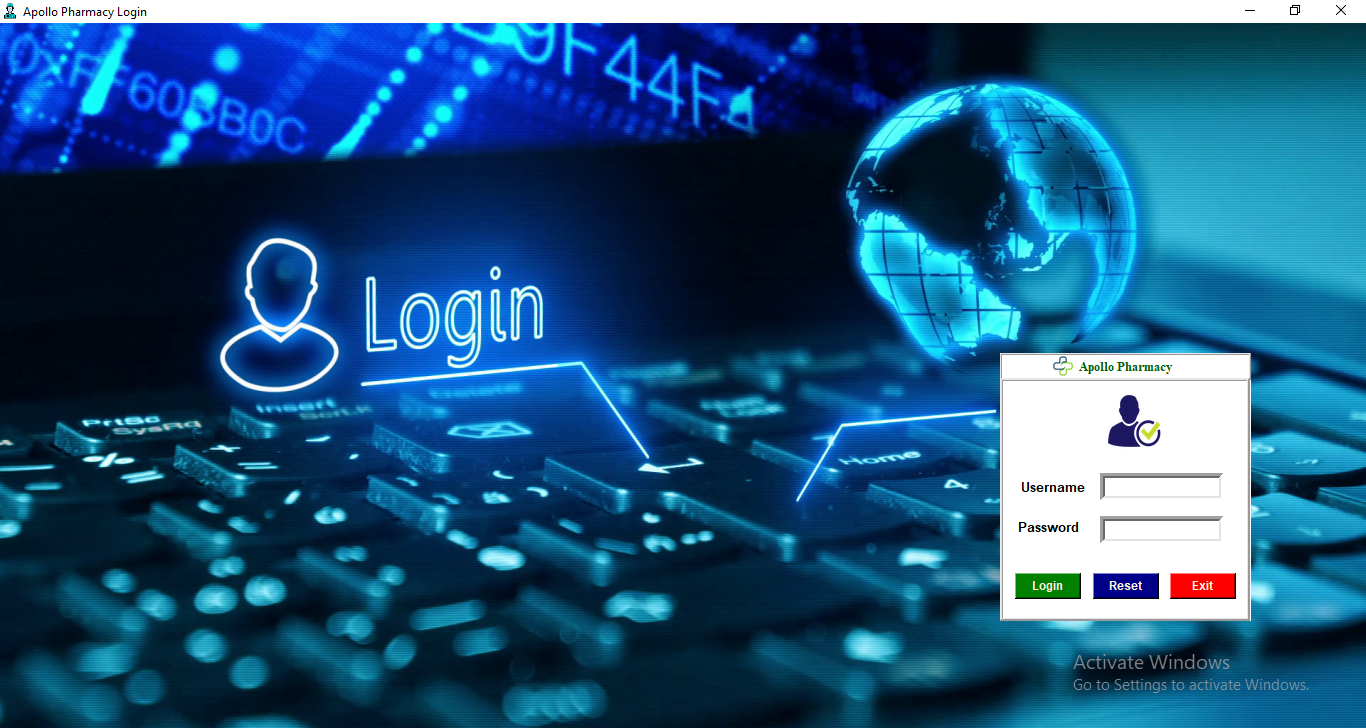


Figure 4.2 Showing Admin login

Figure 4.2 above, shows the admin login area of the application. After a successful login supplying the correct username and password, it opens into another page where the activities of the admin module are fully stipulated. The admin controls all the major activitis of this application. Activities such as Add Drugs, Manage drugs, View sales etc can only be controlled by the admin.

ADMIN MAIN MENU

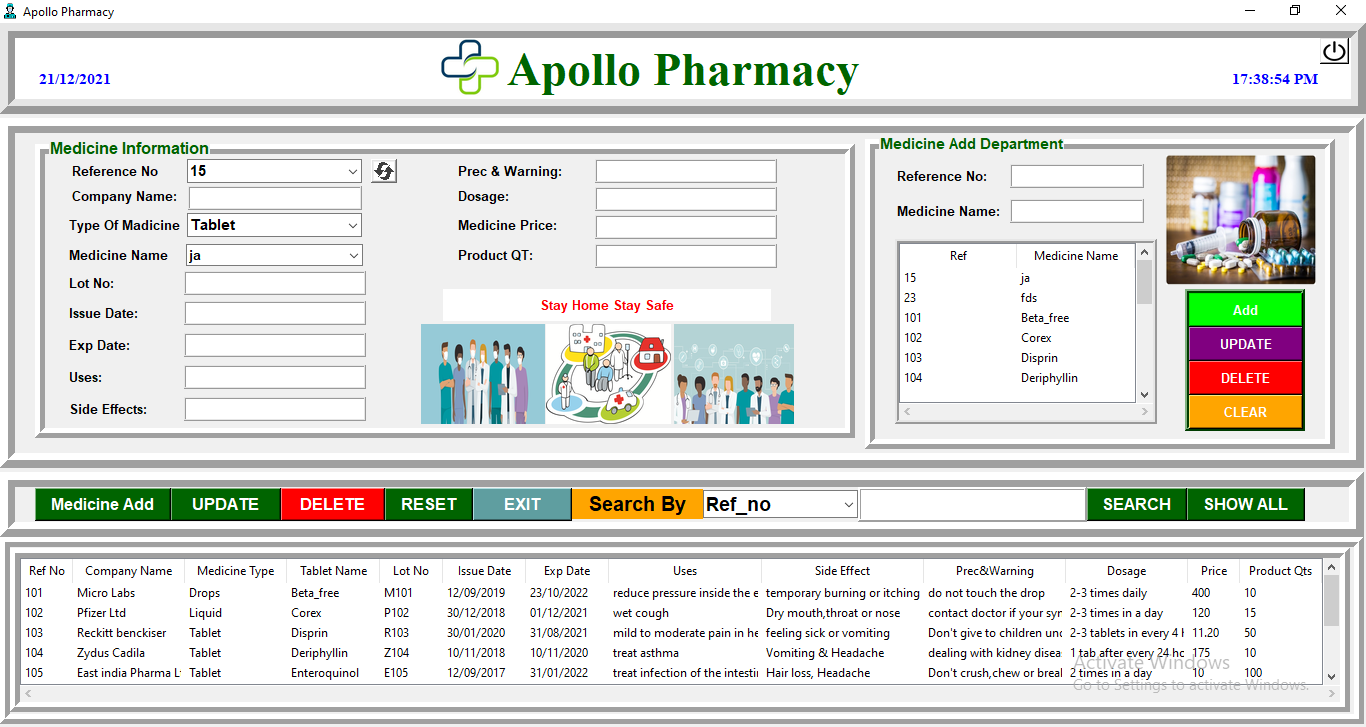


Figure 4.3 Showing Drug Registration Form and Information Registration Form

Figure 4.3 above shows the Drug registration form and Information Registration Form. New drugs brought to the pharmacy are registered here and add their information like uses, expiry, Mfg., Dosages, etc. It receives the details of new batches of drug in the pharmacy.

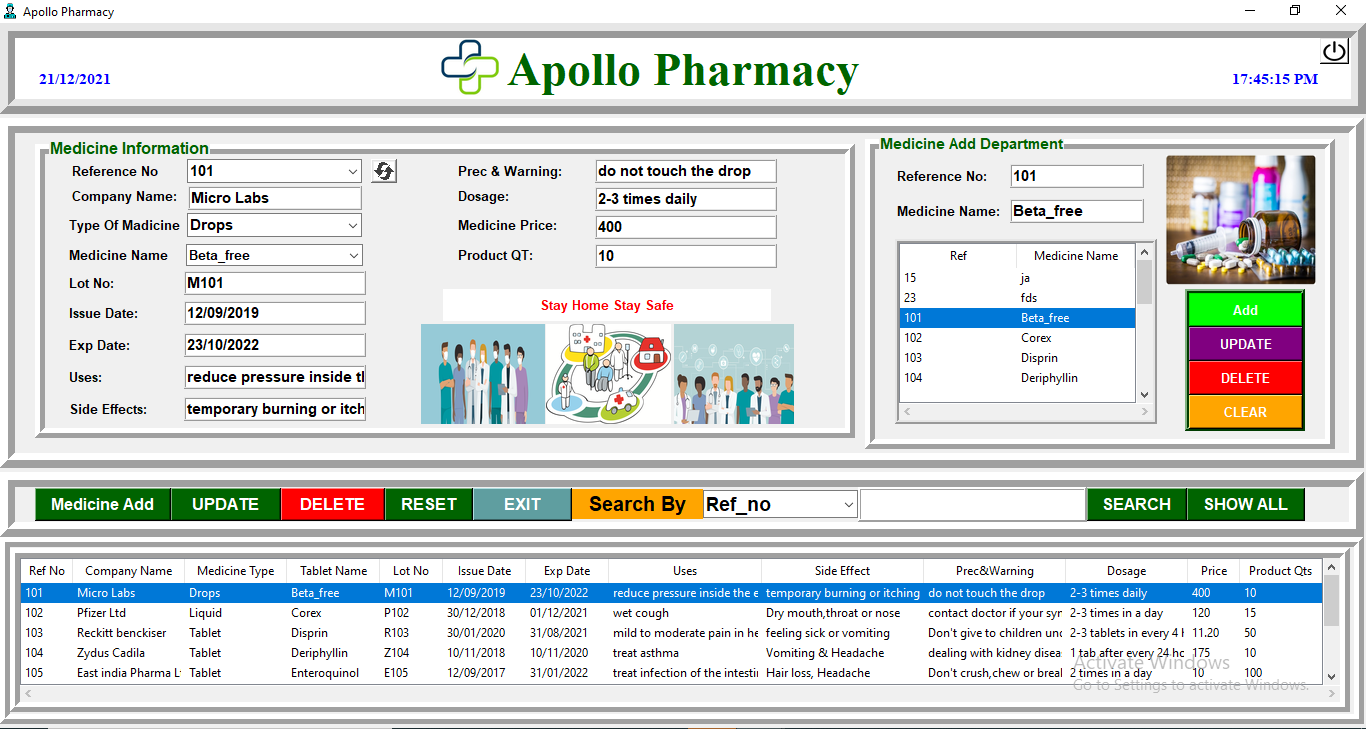
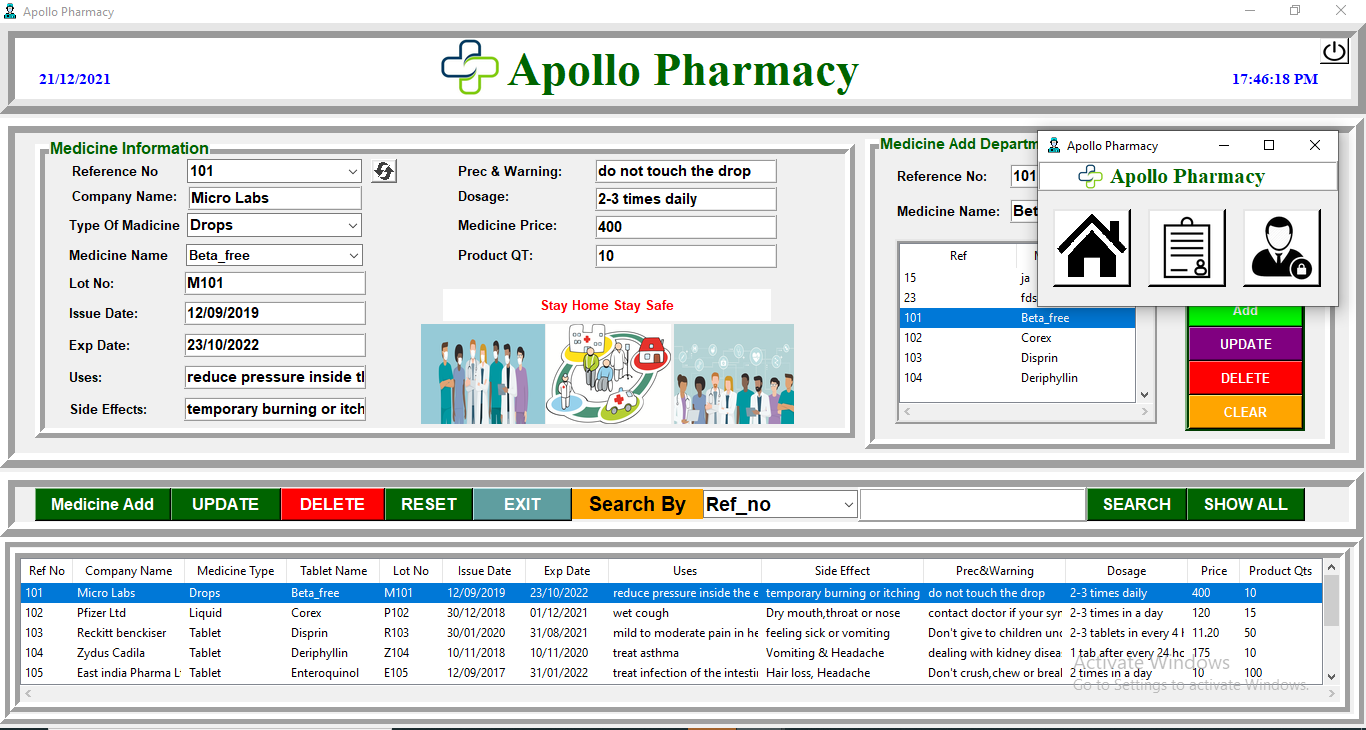


Figure 4.4 ADMIN MAIN MENU WITH ENTRIES

Figure 4.5 ADMIN MAIN MENU WITH POPUP BUTTON MENU

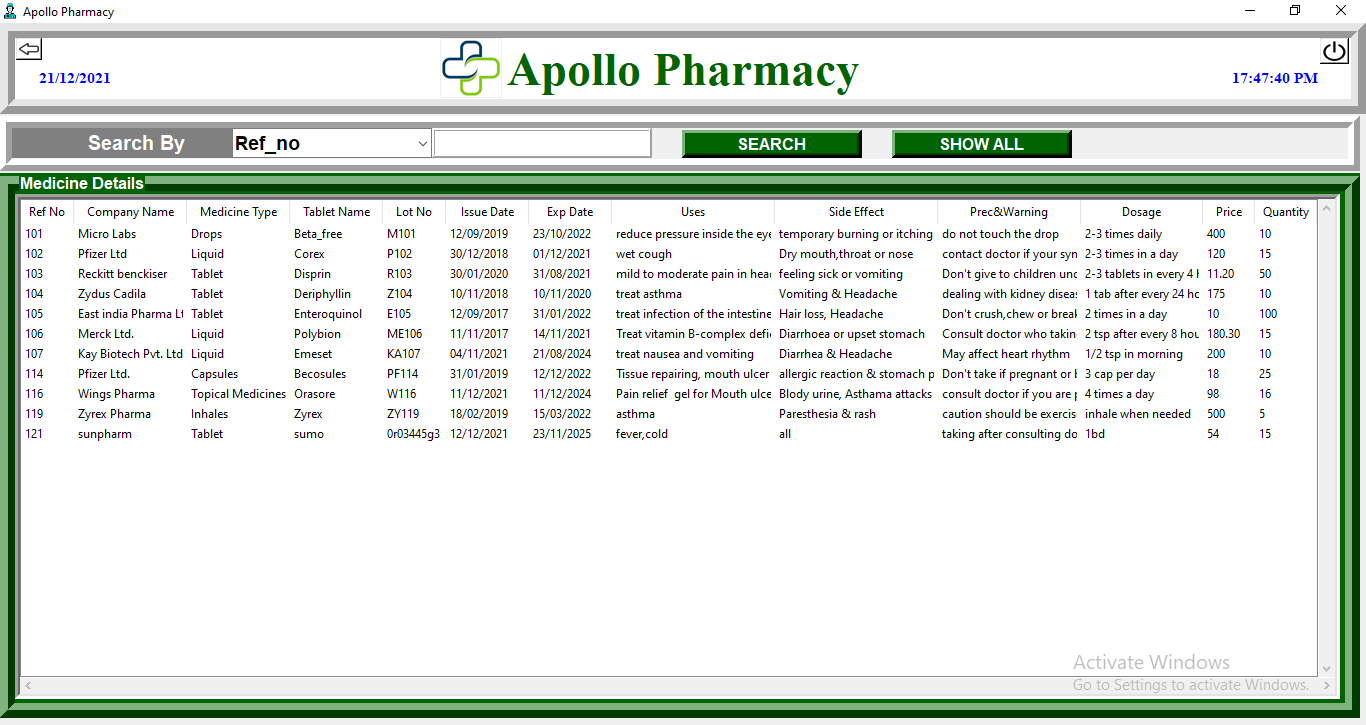
DATABASE WINDOW

Figure 4.6 Showing Medicine details

Figure 4.6 shows the list of drugs in the pharmacy. There is an option to view the drugs in the pharmacy with their information.

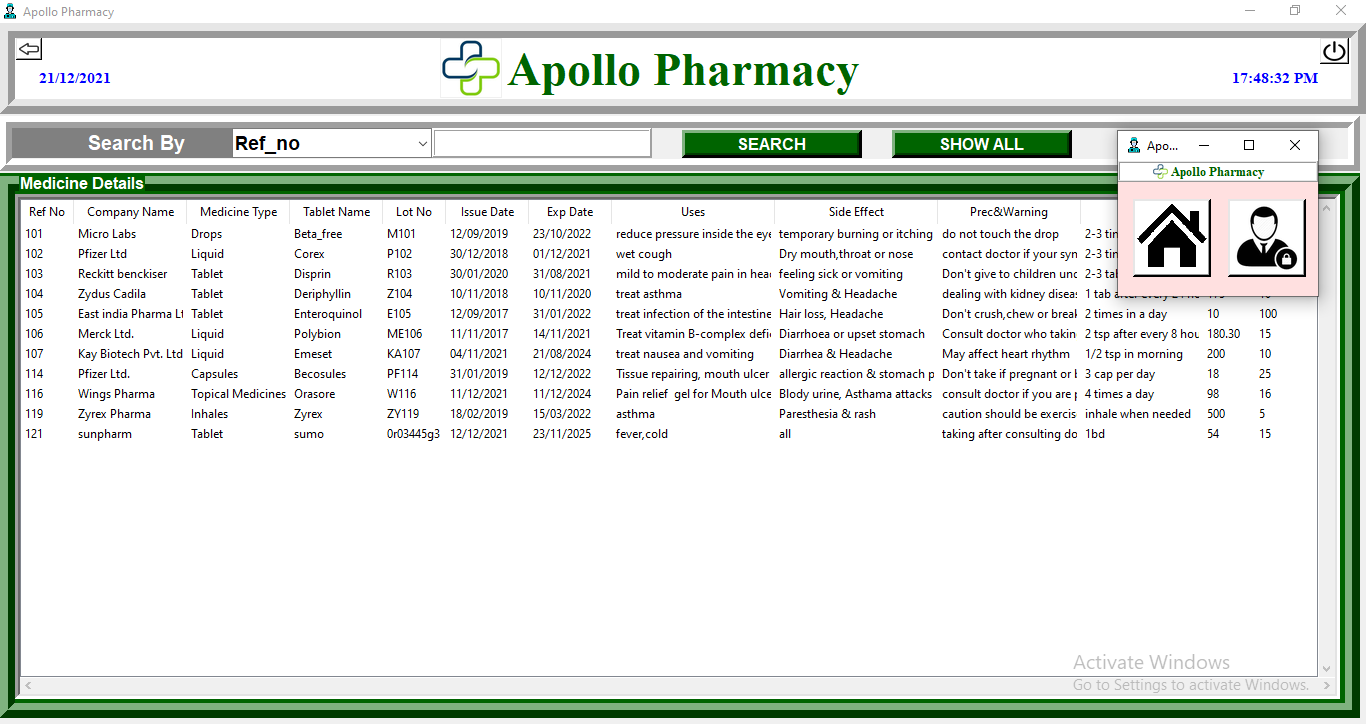


Figure 4.7 DATABASE WINDOW WITH NEW POPUP WINDOW

# 4.8 SYSTEM MAINTENANCE

Maintenance is a continuous process of making modifications and upgrading the application. This usually commences after the application has gone into use. There are two different ways by which this application can be maintained. They include:

* Additive or Enhancement maintenance**:** Business processes are dynamic. As business processes change, applications that support these processes must evolve to reflect these changes. Thus, for this application to perform optimally and to meet changing user requirements, it must be modified continuously.
* Corrective maintenance: This is required in the event that an error occurred when the application is in use. Corrections must be made to changes discovered that can cause malfunctioning of the system.

# **CONCLUSION AND RECOMMENDATION**

# **5.0** SUMMARY

Apollo Pharmacy is designed to improve the accuracy, enhance safety and efficiency in the pharmaceutical store. It is a computer based system which helps the Pharmacist to improve inventory management, cost, medical safety etc.

Apollo Pharmacy was developed to ensure the security of information and reliability of Pharmacy records when accessing and providing services to the customers. The information gathered during the data collection was properly analysed and the results provided the basis for the new system. The system was tested and found to be functional and the outputs produced by this system were encouraging. The application will hence reduce the loss of information unlike the existing system and also information will be processed fast.

# 5.1 CONCLUSION

Effective implementation of this software will take care of the basic requirements of the Apollo pharmacy because it is capable of providing easy and effective storage of information related to activities happening in the stipulated area. With these, the objectives of the system design will be achieved.

In order to allow for future expansion, the system has been designed in such a way that will allow possible modification as it may deem necessary by the pharmacy management, whenever the idea arises.

# 5.2 RECCOMMENDATION

Designing this application (Apollo Pharmacy) is not an easy task. It all started from the requirement gathering and passes through so many other stages before completion.

Based on the benefits of this system and tremendous value it will add to customer-user satisfaction, the below recommendation will be considered;

It is recommended that the new system should be used with the necessary specifications of the system requirements and provision for an uninterrupted power supply should be made available throughout the hours of operation of the pharmacy to avoid power outage. There should also be basic computer knowledge for the users of the software.

It is recommended that the software be improved especially in areas of accounting as it will be of great impact to the development of retail pharmacy

APPENDIX SOURCE CODE

**Main Page**

from datetime import date  
from tkinter import \*  
from PIL import Image, ImageTk  
from time import strftime  
from login import PharmaLogin  
  
  
*# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main page\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**class Main\_page:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Apollo Pharmacy")  
 self.root.geometry("1550x800+0+0")  
 self.root.iconbitmap("photos/Avatar-doctor.ico")  
 self.root.state('zoomed') *# open main\_frame\_window in maximize state.* self.photo = ImageTk.PhotoImage(file='photos/bg22.jpg')  
 lb\_bg = Label(self.root, image=self.photo)  
 lb\_bg.place(x=3, y=0, relwidth=1, relheight=1)  
  
 *# =========================DateTime======================* def time():  
 string = strftime('%I:%M:%S %p')  
 lb1.config(text=string)  
 lb1.after(1000, time)  
  
 lb1 = Label(lb\_bg, font=('times new roman', 11, 'bold'), background='#d8fafb', foreground='darkgreen')  
 lb1.place(x=1050, y=2, width=120, height=40)  
 time()  
  
 today = date.today()  
 string = today.strftime("%a, %d/%b/%Y")  
 lb1.config(text=string)  
  
 lb1 = Label(lb\_bg, font=('times new roman', 11, 'bold'), background='#d8fafb', foreground='darkgreen')  
 lb1.place(x=1200, y=2, width=120, height=40)  
 time()  
  
  
  
  
 lbl\_Title = Label(self.root, text="Apollo Pharmacy", font=("times new roman", 45, "bold"), bg="#d8fafb",  
 fg="darkgreen")  
 lbl\_Title.place(x=840, y=300)  
  
 img\_logo = Image.open("photos/logo3.jpg")  
 img\_logo = img\_logo.resize((80, 80), Image.ANTIALIAS)  
 self.photoimg\_logo = ImageTk.PhotoImage(img\_logo)  
 b1 = Label(self.root, image=self.photoimg\_logo, bg="#d8fafb", borderwidth=0)  
 b1.place(x=760, y=300)  
  
  
 *#===============================================================* img = Image.open("photos/login-button.png")  
 img = img.resize((172, 55), Image.ANTIALIAS)  
 self.photoimg = ImageTk.PhotoImage(img)  
 b2 = Button(self.root, image=self.photoimg, relief=FLAT, borderwidth=1, bg="#d8fafb", command=self.Login\_window)  
 b2.place(x=940, y=430)  
  
  
 img\_med = Image.open("photos/full-med.png")  
 img\_med = img\_med.resize((250, 200), Image.ANTIALIAS)  
 self.photoimg\_med = ImageTk.PhotoImage(img\_med)  
 lbl2 = Label(self.root, image=self.photoimg\_med, bg="#d5f7f8")  
 lbl2.place(x=1120, y=520)  
  
 lbl\_down = Label(self.root, text="Welcome To Apollo Pharmacy", font=("times new roman", 10, "bold"),  
 bg="#d8fafb")  
 lbl\_down.place(x=620, y=655)  
  
 def Login\_window(self):  
 self.root.withdraw()  
 self.LoginWindow = Toplevel(self.root)  
 self.obj = PharmaLogin(self.LoginWindow)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 root = Tk()  
 obj = Main\_page(root)  
 root.mainloop()

**output:**

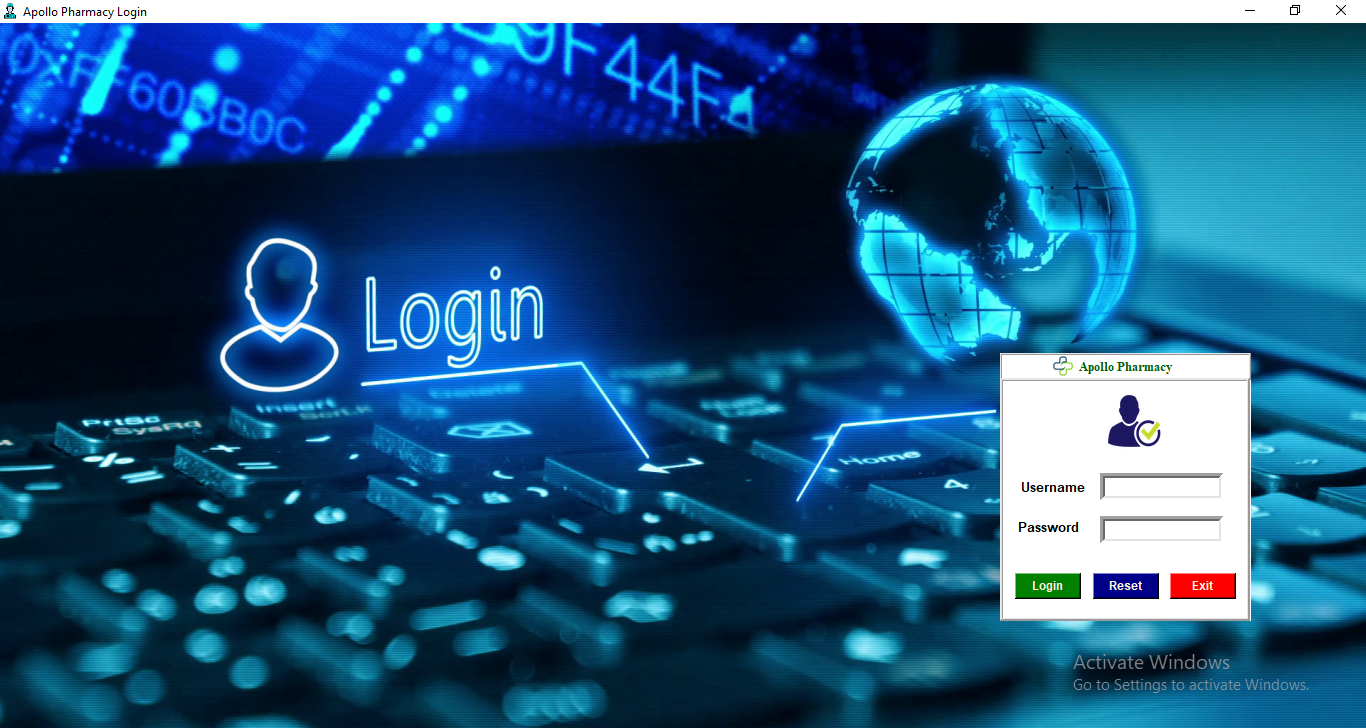


**Login Page**

from tkinter import \*  
from PIL import Image, ImageTk  
from tkinter import messagebox  
from subprocess import call  
  
  
*# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Login\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**class PharmaLogin:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Apollo Pharmacy Login")  
 self.root.geometry("500x600+0+0")  
 self.root.state('zoomed')  
 self.root.config(bg='White')  
 self.frame = Frame(self.root)  
 self.root.iconbitmap('photos/Avatar-doctor.ico')  
 self.frame.pack()  
  
 self.Username = StringVar()  
 self.Password = StringVar()  
  
 self.photo = ImageTk.PhotoImage(file='photos/backlogin.png')  
 lb\_bg = Label(self.root, image=self.photo)  
 lb\_bg.place(x=0, y=0, relwidth=1, relheight=1)  
 *#* login\_frame = Frame(self.root, relief=RIDGE, bg="white")  
 login\_frame.place(x=1000, y=330, width=251, height=268)  
  
 lbltitle = Label(login\_frame, text="Apollo Pharmacy", bd=2, relief=RIDGE  
 , bg="white", fg="darkgreen", font=("times new roman", 12, "bold"), padx=2, pady=4)  
 lbltitle.pack(side=TOP, fill=X)  
  
 img\_logo = Image.open("photos/logo2.jpg")  
 img\_logo = img\_logo.resize((20, 20), Image.ANTIALIAS)  
 self.photoimg1 = ImageTk.PhotoImage(img\_logo)  
 b1 = Button(lbltitle, image=self.photoimg1, borderwidth=0, bg='white')  
 b1.place(x=40, y=0)  
  
 border\_frame = Label(login\_frame, bd=3, relief=RIDGE  
 , bg="white", fg="darkgreen", font=("times new roman", 10, "bold"), padx=122, pady=111)  
 border\_frame.place(x=0, y=25)  
  
 img0 = Image.open("photos/loginphoto.png")  
 img0 = img0.resize((60, 58), Image.ANTIALIAS)  
 self.photoimg2 = ImageTk.PhotoImage(img0)  
 b1 = Button(border\_frame, image=self.photoimg2, borderwidth=0, bg='white')  
 b1.place(x=100, y=10)  
  
 self.lblUsername = Label(login\_frame, text='Username', font=("arial", 10, "bold"), bd=5, bg="white",  
 fg="black")  
 self.lblUsername.place(x=15, y=120)  
  
 self.txtUsername = Entry(login\_frame, font=("arial", 10, "bold"), bd=5, textvariable=self.Username,  
 width=16)  
 self.txtUsername.place(x=100, y=120)  
  
 self.lblPassword = Label(login\_frame, text='Password', font=("arial", 10, "bold"), bd=2, bg="white",  
 fg="black")  
 self.lblPassword.place(x=15, y=163)  
  
 self.txtPassword = Entry(login\_frame, font=("arial", 10, "bold"), show='\*', bd=5,  
 textvariable=self.Password, width=16)  
 self.txtPassword.place(x=100, y=163)

*# ============Buttons================================* self.btnLogin = Button(login\_frame, text='Login', width=8, font=("arial", 9, "bold"), bg="green",  
 fg="cornsilk", command=self.Login\_System)  
 self.btnLogin.place(x=15, y=220)  
  
 self.btnReset = Button(login\_frame, text='Reset', width=8, font=("arial", 9, "bold"), bg="dark blue",  
 fg="cornsilk", command=self.iReset)  
 self.btnReset.place(x=93, y=220)  
  
 self.btnExit = Button(login\_frame, text='Exit', width=8, font=("arial", 9, "bold"), bg="red",  
 fg="cornsilk", command=self.iExit)  
 self.btnExit.place(x=170, y=220)  
  
 *# ==========================================* def Login\_System(self):  
 user = (self.Username.get())  
 pas = (self.Password.get())  
 if (user == str(123)) and pas == str(123):  
 messagebox.showinfo("Login", "Login Successful")  
 self.Login\_window()  
  
 else:  
 messagebox.showwarning("Apollo Login System", "Invalid login Details")  
 self.Username.set("")  
 self.Password.set("")  
 def iReset(self):  
 self.Username.set("")  
 self.Password.set("")  
  
 def iExit(self):  
 self.iExit = messagebox.askyesno("Apollo Login System", "Confirm if you want to exit")  
 if self.iExit > 0:  
 self.root.destroy()  
 return  
  
 def Login\_window(self):  
 self.root.withdraw()  
 call(['python','apollo\_main\_page.py'])  
   
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 root = Tk()  
 obj = PharmaLogin(root)  
 root.mainloop()

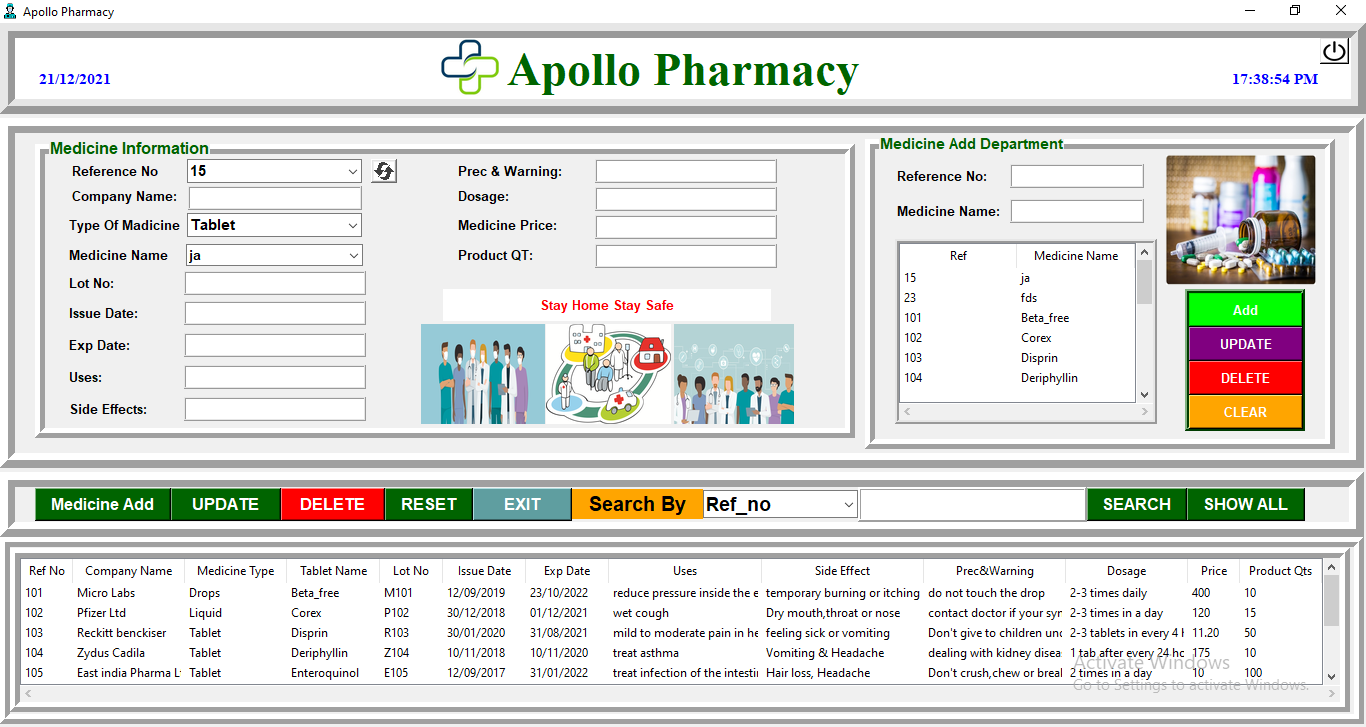
**Output:**

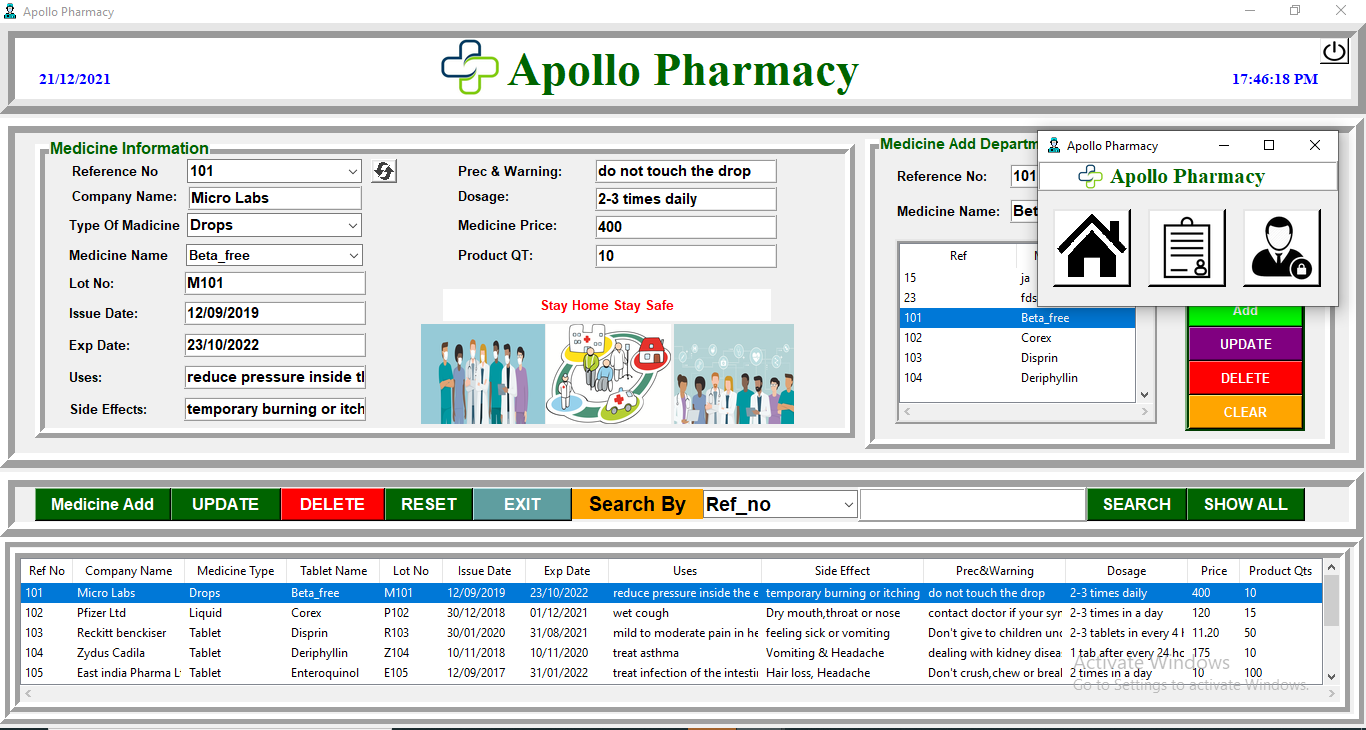


**Drug Registration Form and Information Registration Form**

from tkinter import \*  
from PIL import Image, ImageTk  
from tkinter import ttk  
from tkinter import messagebox  
import sqlite3  
from time import strftime  
from datetime import date  
import tkinter as tk  
from subprocess import call  
  
  
class ApolloPharmacy(tk.Frame):  
  
 def \_\_init\_\_(self,root):  
 tk.Frame.\_\_init\_\_(self,root)  
 self.root = root  
 self.root.title("Apollo Pharmacy")  
 self.root.geometry("1550x800+0+0")  
 self.root.focus()  
 *#self.root.resizable(False,False) # prevent resize of window* self.root.iconbitmap("photos/Avatar-doctor.ico")  
 self.root.state('zoomed') *# open main\_frame\_window in maximize state.  
  
 # ==================addMed variable======================* self.addmed\_var = StringVar()  
 self.refMed\_var = StringVar()  
  
 *# ==================main variable========================* self.ref\_var = StringVar()  
 self.cmpName\_var = StringVar()  
 self.typeMed\_var = StringVar()  
 self.medName\_var = StringVar()  
 self.lot\_var = StringVar()  
 self.issuedate\_var = StringVar()  
 self.expdate\_var = StringVar()  
 self.uses\_var = StringVar()  
 self.sideeffect\_var = StringVar()  
 self.warning\_var = StringVar()  
 self.dosage\_var = StringVar()  
 self.price\_var = StringVar()  
 self.product\_var = StringVar()  
  
 lbltitle = Label(self.root, text="Apollo Pharmacy", bd=15, relief=RIDGE  
 , bg="white", fg="darkgreen", font=("times new roman", 35, "bold"), padx=2, pady=4)  
 lbltitle.pack(side=TOP, fill=X)  
  
 *# img1 = Image.open("C:\\Users\\bhanu\\OneDrive\\Desktop\\pharma image\\Bhanu-logo.jpg")* img1 = Image.open("photos/logo2.jpg")  
 img1 = img1.resize((60, 58), Image.ANTIALIAS)  
 self.photoimg1 = ImageTk.PhotoImage(img1)  
 b1 = Label(self.root, image=self.photoimg1, borderwidth=0)  
 b1.place(x=440, y=15)  
  
  
 img\_login\_back = Image.open("photos/logout.jpg")  
 img\_login\_back = img\_login\_back.resize((23, 20), Image.ANTIALIAS)  
 self.photoimg12 = ImageTk.PhotoImage(img\_login\_back)  
 bt\_login\_back = Button(self.root, image=self.photoimg12, relief=RAISED, borderwidth=2,bg="white",command=self.LogOut\_w)  
 bt\_login\_back.place(x=1320, y=15)  
  
 *# =========================DateTime======================* def time():  
 string = strftime('%H:%M:%S %p')  
 lb1.config(text=string)  
 lb1.after(1000, time)  
  
 lb1 = Label(lbltitle, font=('times new roman', 11, 'bold'), background='white', foreground='blue')  
 lb1.place(x=0, y=21, width=120, height=40)  
 time()  
  
 today = date.today()  
 string = today.strftime("%d/%m/%Y")  
 lb1.config(text=string)  
  
 lb1 = Label(lbltitle, font=('times new roman', 11, 'bold'), background='white', foreground='blue')  
 lb1.place(x=1200, y=21, width=120, height=40)  
 time()  
  
 *# ===================DataFrame==========================* DataFrame = Frame(self.root, bd=15, relief=RIDGE, padx=20)  
 DataFrame.place(x=0, y=95, width=1364, height=350)  
  
 DataFrameleft = LabelFrame(DataFrame, bd=10, relief=RIDGE, padx=20, text="Medicine Information", fg="darkgreen",  
 font=("arial", 12, "bold"))  
 DataFrameleft.place(x=0, y=5, width=820, height=300)  
  
 *# =============ButtonsFrames==============================* ButtonFrame = Frame(self.root, bd=15, relief=RIDGE, padx=20)  
 ButtonFrame.place(x=0, y=449, width=1364, height=65)  
  
 *#========================RefreshButton========================* img\_refresh = Image.open("photos/refresh\_btn.png")  
 img\_refresh = img\_refresh.resize((20, 18), Image.ANTIALIAS)  
 self.photoimg9 = ImageTk.PhotoImage(img\_refresh)  
 bt\_refresh = Button(self.root, image=self.photoimg9, relief=RAISED, borderwidth=2, command=self.refresh\_w)  
 bt\_refresh.place(x=371, y=136)  
 *#  
 # img\_search = Image.open("photos/search\_btn.png")  
 # img\_search = img\_search.resize((20, 18), Image.ANTIALIAS)  
 # self.photoimg10 = ImageTk.PhotoImage(img\_search)  
 # bt\_search = Button(self.root, image=self.photoimg10, relief=RAISED, borderwidth=2, command=self.search\_data1)  
 # bt\_search.place(x=401, y=136)  
  
 # =====================Main Button========================* btnAddData = Button(ButtonFrame, text="Medicine Add", padx=10, font=("arial", 12, "bold"), bg="darkgreen",  
 fg="white",  
 command=self.add\_data)  
 btnAddData.grid(row=0, column=0)  
  
 btnUpdateMed = Button(ButtonFrame, text="UPDATE", padx=15, font=("arial", 13, "bold"), bg="darkgreen",  
 fg="white", command=self.Update)  
 btnUpdateMed.grid(row=0, column=1)  
  
 btnDeleteMed = Button(ButtonFrame, text="DELETE", padx=13, font=("arial", 13, "bold"), bg="red", fg="white",  
 command=self.delete)  
 btnDeleteMed.grid(row=0, column=2)  
  
 btnRestMed = Button(ButtonFrame, text="RESET", padx=10, font=("arial", 13, "bold"), bg="darkgreen", fg="white",  
 command=self.reset)  
 btnRestMed.grid(row=0, column=3)  
  
 btnExitMed = Button(ButtonFrame, text="EXIT", padx=25, font=("arial", 13, "bold"), bg="cadet blue", fg="white",  
 command=self.Quit)  
 btnExitMed.grid(row=0, column=4)  
  
 *# ==================Search BY=====================* lblSearch = Label(ButtonFrame, font=("arial", 15, "bold"), text="Search By", padx=15, bg="orange", fg="Black")  
 lblSearch.grid(row=0, column=5, sticky=W)  
  
 *# variable* self.search\_var = StringVar()  
 search\_combo = ttk.Combobox(ButtonFrame, textvariable=self.search\_var, width=12, font=("arial", 14, "bold"),  
 state="readonly")  
 search\_combo["values"] = ("Ref\_no", "medName", "lotNo")  
 search\_combo.grid(row=0, column=6)  
 search\_combo.current(0)  
  
 *# variable2* self.searchTxt\_var = StringVar()  
 txtSearch = Entry(ButtonFrame, textvariable=self.searchTxt\_var, bd=3, relief=RIDGE, width=17,  
 font=("arial", 17, "bold"))  
 txtSearch.grid(row=0, column=7)  
  
 searchBtn = Button(ButtonFrame, text="SEARCH", padx=9, font=("arial", 12, "bold"), bg="darkgreen", fg="white",  
 command=self.search\_data)  
 searchBtn.grid(row=0, column=8)  
  
 showAll = Button(ButtonFrame, text="SHOW ALL", padx=10, font=("arial", 12, "bold"), bg="darkgreen", fg="white",  
 command=self.fetch\_data)  
 showAll.grid(row=0, column=9)  
  
 *# =======================label and entry=================* lblrefno = Label(DataFrameleft, font=("arial", 10, "bold"), text="Reference No", padx=5)  
 lblrefno.grid(row=0, column=0, sticky=W)  
  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("select Ref from pharma")  
 row = my\_cursor.fetchall()  
  
 *# self.searchTxt\_var1 = StringVar()  
 # ref\_combo = ttk.Combobox(DataFrameleft,textvariable=self.searchTxt\_var1, width=19, font=("arial", 11, "bold"),  
 # state="readonly")  
 # ref\_combo["values"] = row  
 # ref\_combo.grid(row=0, column=1)  
 # ref\_combo.current(0)* ref\_combo = ttk.Combobox(DataFrameleft, textvariable=self.ref\_var, width=19, font=("arial", 11, "bold"),  
 state="readonly")  
 ref\_combo["values"] = row  
 ref\_combo.grid(row=0, column=1)  
 ref\_combo.current(0)  
  
 lblCmpName = Label(DataFrameleft, font=("arial", 10, "bold"), text="Company Name:", pady=1)  
 lblCmpName.grid(row=1, column=0)  
  
 self.txtcmpName = Entry(DataFrameleft, textvariable=self.cmpName\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=21)  
 self.txtcmpName.place(x=123,y=27)  
  
  
 lblTypeofMedicine = Label(DataFrameleft, font=("arial", 10, "bold"), text="Type Of Madicine", padx=2, pady=6)  
 lblTypeofMedicine.grid(row=2, column=0, sticky=W)  
  
 comTypeofMedicine = ttk.Combobox(DataFrameleft, textvariable=self.typeMed\_var, state="readonly",  
 font=("arial", 11, "bold"), width=19)  
 comTypeofMedicine["value"] = (  
 "Tablet", "Liquid", "Capsules", "Topical Medicines", "Drops", "Inhales", "Injection")  
 comTypeofMedicine.current(0)  
 comTypeofMedicine.grid(row=2, column=1)  
  
 *# ================Add Medicine============================* lblMedicineName = Label(DataFrameleft, font=("arial", 10, "bold"), text="Medicine Name", padx=2, pady=4)  
 lblMedicineName.grid(row=3, column=0, sticky=W)  
  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("select MedName from pharma")  
 med = my\_cursor.fetchall()  
  
 comMedicineName = ttk.Combobox(DataFrameleft, textvariable=self.medName\_var, state="readonly",  
 font=("arial", 10, "bold"), width=22)  
 comMedicineName["value"] = med  
 comMedicineName.current(0)  
 comMedicineName.grid(row=3, column=1)  
  
 lblLotNo = Label(DataFrameleft, font=("arial", 10, "bold"), text="Lot No:", padx=2, pady=4)  
 lblLotNo.grid(row=4, column=0, sticky=W)  
 txtLotNo = Entry(DataFrameleft, textvariable=self.lot\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=22)  
 txtLotNo.grid(row=4, column=1)  
  
 lblIssueDate = Label(DataFrameleft, font=("arial", 10, "bold"), text="Issue Date:", padx=2, pady=6)  
 lblIssueDate.grid(row=5, column=0, sticky=W)  
 txtIssueDate = Entry(DataFrameleft, textvariable=self.issuedate\_var, font=("arial", 11, "bold"), bg="white",  
 bd=2, relief=RIDGE, width=22)  
 txtIssueDate.grid(row=5, column=1)  
  
 lblExDate = Label(DataFrameleft, font=("arial", 10, "bold"), text="Exp Date:", padx=2, pady=6)  
 lblExDate.grid(row=6, column=0, sticky=W)  
 txtExDate = Entry(DataFrameleft, textvariable=self.expdate\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=22)  
 txtExDate.grid(row=6, column=1)  
  
 lblUses = Label(DataFrameleft, font=("arial", 10, "bold"), text="Uses:", padx=2, pady=6)  
 lblUses.grid(row=7, column=0, sticky=W)  
 txtUses = Entry(DataFrameleft, textvariable=self.uses\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=22)  
 txtUses.grid(row=7, column=1)  
  
 lblSideEffect = Label(DataFrameleft, font=("arial", 10, "bold"), text="Side Effects:", padx=2, pady=6)  
 lblSideEffect.grid(row=8, column=0, sticky=W)  
 txtSideEffect = Entry(DataFrameleft, textvariable=self.sideeffect\_var, font=("arial", 11, "bold"), bg="white",  
 bd=2, relief=RIDGE, width=22)  
 txtSideEffect.grid(row=8, column=1)  
 *# =====================================================* lblPrecWarning = Label(DataFrameleft, font=("arial", 10, "bold"), text="Prec & Warning:", padx=90, pady=1)  
 lblPrecWarning.grid(row=0, column=2, sticky=W)  
 txtPrecWarning = Entry(DataFrameleft, textvariable=self.warning\_var, font=("arial", 11, "bold"), bg="white",  
 bd=2, relief=RIDGE, width=22)  
 txtPrecWarning.place(x=530,y=0)  
  
 lblDosage = Label(DataFrameleft, font=("arial", 10, "bold"), text="Dosage:", padx=90, pady=3)  
 lblDosage.grid(row=1, column=2, sticky=W)  
 txtDosage = Entry(DataFrameleft, textvariable=self.dosage\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=22)  
 txtDosage.place(x=530,y=28)  
  
 lblPrice = Label(DataFrameleft, font=("arial", 10, "bold"), text="Medicine Price:", padx=90, pady=3)  
 lblPrice.grid(row=2, column=2, sticky=W)  
 txtPrice = Entry(DataFrameleft, textvariable=self.price\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=22)  
 txtPrice.place(x=530,y=56)  
  
 lblProductQt = Label(DataFrameleft, font=("arial", 10, "bold"), text="Product QT:", padx=90, pady=3)  
 lblProductQt.grid(row=3, column=2, sticky=W)  
 txtProductQt = Entry(DataFrameleft, textvariable=self.product\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=22)  
 txtProductQt.place(x=530, y=85)  
  
  
 *# ============================Images=======================* lblhome = Label(DataFrameleft, font=("arial", 10, "bold"), text="Stay Home Stay Safe", bg="white", fg="red",  
 width=40, padx=2, pady=6)  
 lblhome.place(x=378, y=130)  
  
 img2 = Image.open("photos/Doctors.jpg")  
 img2 = img2.resize((125, 100), Image.ANTIALIAS)  
 self.photoimg2 = ImageTk.PhotoImage(img2)  
 b1 = Button(self.root, image=self.photoimg2, borderwidth=0)  
 b1.place(x=420, y=300)  
  
 img3 = Image.open("photos/doctor2.png")  
 img3 = img3.resize((125, 100), Image.ANTIALIAS)  
 self.photoimg3 = ImageTk.PhotoImage(img3)  
 b1 = Button(self.root, image=self.photoimg3, borderwidth=0)  
 b1.place(x=545, y=300)  
  
 img4 = Image.open("photos/Doctors1.jpg")  
 img4 = img4.resize((120, 100), Image.ANTIALIAS)  
 self.photoimg4 = ImageTk.PhotoImage(img4)  
 b1 = Button(self.root, image=self.photoimg4, borderwidth=0)  
 b1.place(x=673, y=300)  
  
 *# =================== dataframeRight======================* DataFrameright = LabelFrame(DataFrame, bd=10, relief=RIDGE, padx=20, text="Medicine Add Department",  
 fg="darkgreen", font=("arial", 11, "bold"))  
 DataFrameright.place(x=830, y=1, width=470, height=315)  
  
 img7 = Image.open("photos/mediadd.jpg")  
 img7 = img7.resize((150, 130), Image.ANTIALIAS)  
 self.photoimg7 = ImageTk.PhotoImage(img7)  
 b1 = Button(self.root, image=self.photoimg7, borderwidth=0)  
 b1.place(x=1165, y=131)  
 *# ============================================* lblrefno = Label(DataFrameright, font=("arial", 10, "bold"), text="Reference No:", padx=0, pady=2)  
 lblrefno.place(x=0, y=10)  
 txtrefno = Entry(DataFrameright, textvariable=self.refMed\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=16)  
 txtrefno.place(x=115, y=10)  
  
  
 lblmedName = Label(DataFrameright, font=("arial", 10, "bold"), text="Medicine Name:", padx=0, pady=2)  
 lblmedName.place(x=0, y=45)  
 txtmedName = Entry(DataFrameright, textvariable=self.addmed\_var, font=("arial", 11, "bold"), bg="white", bd=2,  
 relief=RIDGE, width=16)  
 txtmedName.place(x=115, y=45)  
  
 *# =======================Side frame======================* side\_frame = Frame(DataFrameright, bd=4, relief=RIDGE, bg="white")  
 side\_frame.place(x=0, y=85, width=262, height=185)  
  
 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)  
  
 self.medicine\_table = ttk.Treeview(side\_frame, column=("ref", "medname"), xscrollcommand=sc\_x.set,  
 yscrollcommand=sc\_y.set)  
 sc\_x.config(command=self.medicine\_table.xview)  
 sc\_y.config(command=self.medicine\_table.yview)  
  
 self.medicine\_table.heading("ref", text="Ref")  
 self.medicine\_table.heading("medname", text="Medicine Name")  
  
 self.medicine\_table["show"] = "headings"  
 self.medicine\_table.pack(fill=BOTH, expand=1)  
  
 self.medicine\_table.column("ref", width=100)  
 self.medicine\_table.column("medname", width=100)  
  
 self.medicine\_table.bind("<ButtonRelease-1>", self.Medget\_cursor)  
  
 *# ==================Medicine Add Buttons==================* down\_frame = Frame(DataFrameright, bd=4, relief=RIDGE, bg="darkgreen")  
 down\_frame.place(x=290, y=135, width=120, height=142)  
  
 btnAddmed = Button(down\_frame, text="Add", font=("arial", 10, "bold"), width=13, bg="lime", fg="white", pady=4,  
 command=self.AddMed)  
 btnAddmed.grid(row=0, column=0)  
  
 btnUpdatemed = Button(down\_frame, text="UPDATE", font=("arial", 10, "bold"), width=13, bg="purple", fg="white",  
 pady=4, command=self.UpdateMed)  
 btnUpdatemed.grid(row=1, column=0)  
  
 btnDeletemed = Button(down\_frame, text="DELETE", font=("arial", 10, "bold"), width=13, bg="red", fg="white",  
 pady=4, command=self.DeleteMed)  
 btnDeletemed.grid(row=2, column=0)  
  
 btnClearmed = Button(down\_frame, text="CLEAR", font=("arial", 10, "bold"), width=13, bg="orange", fg="white",  
 pady=4, command=self.ClearMed)  
 btnClearmed.grid(row=3, column=0)  
  
 *# ====================Frame Details===============* Framedetails = Frame(self.root, bd=10, relief=RIDGE)  
 Framedetails.place(x=0, y=514, width=1364, height=187)  
  
 *# ==================== Maximize Button===================  
 # img8 = Image.open("photos/maximize.png")  
 # img8 = img8.resize((28, 26), Image.ANTIALIAS)  
 # self.photoimg8 = ImageTk.PhotoImage(img8)  
 # b2 = Button(self.root, image=self.photo8, relief=RAISED, borderwidth=3, command=self.Max\_Window)  
 # b2.place(x=1323, y=516)  
  
 # =================Main Table & scrollbar ===============* Table\_frame = Frame(Framedetails, bd=10, relief=RIDGE)  
 Table\_frame.place(x=0, y=1, width=1340, height=164)  
  
 scroll\_x = ttk.Scrollbar(Table\_frame, orient=HORIZONTAL)  
 scroll\_x.pack(side=BOTTOM, fill=X)  
 scroll\_y = ttk.Scrollbar(Table\_frame, orient=VERTICAL)  
 scroll\_y.pack(side=RIGHT, fill=Y)  
  
 self.pharmacy\_table = ttk.Treeview(Table\_frame, column=("reg", "companyname", "type", "tabletname", "lotno",  
 "issuedate", "expdate", "uses", "sideeffect", "warning",  
 "dosage",  
 "price", "productqt"), xscrollcommand=scroll\_x.set,  
 yscrollcommand=scroll\_y.set)  
 scroll\_x.pack(side=BOTTOM, fill=X)  
 scroll\_y.pack(side=RIGHT, fill=Y)  
  
 scroll\_x.config(command=self.pharmacy\_table.xview)  
 scroll\_y.config(command=self.pharmacy\_table.yview)  
  
 self.pharmacy\_table["show"] = "headings"  
  
 self.pharmacy\_table.heading("reg", text="Ref No")  
 self.pharmacy\_table.heading("companyname", text="Company Name")  
 self.pharmacy\_table.heading("type", text="Medicine Type")  
 self.pharmacy\_table.heading("tabletname", text="Tablet Name")  
 self.pharmacy\_table.heading("lotno", text="Lot No")  
 self.pharmacy\_table.heading("issuedate", text="Issue Date")  
 self.pharmacy\_table.heading("expdate", text="Exp Date")  
 self.pharmacy\_table.heading("uses", text="Uses")  
 self.pharmacy\_table.heading("sideeffect", text="Side Effect")  
 self.pharmacy\_table.heading("warning", text="Prec&Warning")  
 self.pharmacy\_table.heading("dosage", text="Dosage")  
 self.pharmacy\_table.heading("price", text="Price")  
 self.pharmacy\_table.heading("productqt", text="Product Qts")  
 self.pharmacy\_table.pack(fill=BOTH, expand=1)  
  
 self.pharmacy\_table.column("reg", width=40)  
 self.pharmacy\_table.column("companyname", width=100)  
 self.pharmacy\_table.column("type", width=90)  
 self.pharmacy\_table.column("tabletname", width=80)  
 self.pharmacy\_table.column("lotno", width=50)  
 self.pharmacy\_table.column("issuedate", width=70)  
 self.pharmacy\_table.column("expdate", width=70)  
 self.pharmacy\_table.column("uses", width=140)  
 self.pharmacy\_table.column("sideeffect", width=150)  
 self.pharmacy\_table.column("warning", width=130)  
 self.pharmacy\_table.column("dosage", width=110)  
 self.pharmacy\_table.column("price", width=40)  
 self.pharmacy\_table.column("productqt", width=70)  
 self.fetch\_dataMed()  
 self.fetch\_data()  
 self.pharmacy\_table.bind("<ButtonRelease-1>", self.get\_cursor)  
  
 *# ========Add Medicine Functionality Declaration========* def AddMed(self):  
 if self.refMed\_var.get() == "":  
 messagebox.showerror("Error", "All fields are required")  
 else:  
 try:  
 conn = sqlite3.connect("mydata.db")  
 conn.execute("CREATE TABLE IF NOT EXISTS pharma(Ref INTEGER PRIMARY KEY, MedName TEXT)")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("insert into pharma(Ref,MedName) values(?,?)", (  
 self.refMed\_var.get(),  
 self.addmed\_var.get()  
 ))  
 conn.commit()  
 self.fetch\_dataMed()  
 *# self.Medget\_cursor()* messagebox.showinfo("Success", "Medicine Added")  
 conn.close()  
 except Exception as ex:  
 messagebox.showinfo("Fail", ex)  
  
 def fetch\_dataMed(self):  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("select \* from pharma")  
 rows = my\_cursor.fetchall()  
 if len(rows) != 0:  
 self.medicine\_table.delete(\*self.medicine\_table.get\_children())  
 for i in rows:  
 self.medicine\_table.insert("", END, values=i)  
 conn.commit()  
 conn.close()  
  
 *# ===============MedGetcursor==========================* def Medget\_cursor(self, event=""):  
 cursor\_row = self.medicine\_table.focus()  
 content = self.medicine\_table.item(cursor\_row)  
 row = content["values"]  
 self.refMed\_var.set(row[0])  
 self.addmed\_var.set(row[1])  
  
 def UpdateMed(self):  
 if self.refMed\_var.get() == "" or self.addmed\_var.get() == "":  
 messagebox.showerror("Error", "All fields are Required")  
 else:  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("update pharma set MedName=? where Ref=?", (  
 self.addmed\_var.get(),  
 self.refMed\_var.get(),  
 ))  
 conn.commit()  
 self.fetch\_dataMed()  
  
 conn.close()  
 messagebox.showinfo("success", "Medicine has been Updated")  
  
 def DeleteMed(self):  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
  
 sql = "delete from pharma where Ref=?"  
 val = (self.refMed\_var.get(),)  
 my\_cursor.execute(sql, val)  
  
 conn.commit()  
 self.fetch\_dataMed()  
 conn.close()  
  
 def ClearMed(self):  
 self.refMed\_var.set("")  
 self.addmed\_var.set("")  
  
 *# ==========================Main table=======================* def add\_data(self):  
 if self.ref\_var.get() == "" or self.lot\_var.get() == "":  
 messagebox.showerror("Error", "All fields are required")  
 else:  
 conn = sqlite3.connect("mydata.db")  
 conn.execute(  
 "CREATE TABLE IF NOT EXISTS pharmacy(Ref\_no INTEGER PRIMARY KEY, cmpName varchar(20),TypeMed varchar(20),medName varchar(20), lotNo varchar(20),Issuedate varchar(20), Expdate varchar(20), uses varchar(20),sideeffect varchar(20),warning varchar(20),dosage varchar(20),Price varchar(20),product varchar(20) )")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute(  
 "insert into pharmacy values(?,?,?,?,?,?,?,?,?,?,?,?,?)", (  
 self.ref\_var.get(),  
 self.cmpName\_var.get(),  
 self.typeMed\_var.get(),  
 self.medName\_var.get(),  
 self.lot\_var.get(),  
 self.issuedate\_var.get(),  
 self.expdate\_var.get(),  
 self.uses\_var.get(),  
 self.sideeffect\_var.get(),  
 self.warning\_var.get(),  
 self.dosage\_var.get(),  
 self.price\_var.get(),  
 self.product\_var.get(),  
 ))  
 conn.commit()  
 self.fetch\_data()  
 conn.close()  
 messagebox.showinfo("success", "data has been inserted")  
  
 def fetch\_data(self):  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("select \* from pharmacy")  
 row = my\_cursor.fetchall()  
 if len(row) != 0:  
 self.pharmacy\_table.delete(\*self.pharmacy\_table.get\_children())  
 for i in row:  
 self.pharmacy\_table.insert("", END, values=i)  
 conn.commit()  
 conn.close()  
  
  
 def get\_cursor(self, event=""):  
 cursor\_row = self.pharmacy\_table.focus()  
 content = self.pharmacy\_table.item(cursor\_row)  
 row = content["values"]  
 self.ref\_var.set(row[0]),  
 self.cmpName\_var.set(row[1]),  
 self.typeMed\_var.set(row[2]),  
 self.medName\_var.set(row[3]),  
 self.lot\_var.set(row[4]),  
 self.issuedate\_var.set(row[5]),  
 self.expdate\_var.set(row[6]),  
 self.uses\_var.set(row[7]),  
 self.sideeffect\_var.set(row[8]),  
 self.warning\_var.set(row[9]),  
 self.dosage\_var.set(row[10]),  
 self.price\_var.set(row[11]),  
 self.product\_var.set(row[12])  
  
 def Update(self):  
 if self.ref\_var.get() == "" or self.lot\_var.get() == "":  
 messagebox.showerror("Error", "All fields are Required")  
 else:  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute(  
 "update pharmacy set cmpName=?,TypeMed=?,medName=?, lotNo=?,Issuedate=?,Expdate=?,uses=?,sideeffect=?,warning=?,dosage=?,Price=?,product=? where Ref\_no=?",  
 (  
  
 self.cmpName\_var.get(),  
 self.typeMed\_var.get(),  
 self.medName\_var.get(),  
 self.lot\_var.get(),  
 self.issuedate\_var.get(),  
 self.expdate\_var.get(),  
 self.uses\_var.get(),  
 self.sideeffect\_var.get(),  
 self.warning\_var.get(),  
 self.dosage\_var.get(),  
 self.price\_var.get(),  
 self.product\_var.get(),  
 self.ref\_var.get(),  
 ))  
 conn.commit()  
 self.fetch\_data()  
 conn.close()  
  
 messagebox.showinfo("UPDATE", "Record has been Updated successfully")  
  
 def delete(self):  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
  
 sql = "delete from pharmacy where Ref\_no=?"  
 val = (self.ref\_var.get(),)  
 my\_cursor.execute(sql, val)  
  
 conn.commit()  
 self.fetch\_data()  
 conn.close()  
  
 messagebox.showinfo("DELETE", "Info deleted successfully")  
  
 def reset(self):  
 *# self.ref\_var.set(""),# if we want to show ref than uncomment* self.cmpName\_var.set(""),  
 *# self.typeMed\_var.set(""),  
 # self.medName\_var.set(""),* self.lot\_var.set(""),  
 self.issuedate\_var.set(""),  
 self.expdate\_var.set(""),  
 self.uses\_var.set(""),  
 self.sideeffect\_var.set(""),  
 self.warning\_var.set(""),  
 self.dosage\_var.set(""),  
 self.price\_var.set(""),  
 self.product\_var.set(""),  
 self.ref\_var.set(""),  
  
 def Quit(self):  
 self.destroy = messagebox.askyesno("Apollo Login System", "Confirm if you want to exit")  
 raise SystemExit  
  
 *# def quit(self):  
 # self.tk.quit()  
 #self.root.destroy()  
  
 # def destroy(self):  
 # self.destroy = messagebox.askyesno("Apollo Pharmacy", "Confirm if you want to exit")  
 # if self.destroy > 0:  
 # self.root.destroy()  
 # return* def search\_data(self):  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("select \* from pharmacy where " + str(  
 self.search\_var.get() + " LIKE '" + str(self.searchTxt\_var.get()) + "%'"))  
 rows = my\_cursor.fetchall()  
 if len(rows) != 0:  
 self.pharmacy\_table.delete(\*self.pharmacy\_table.get\_children())  
 for i in rows:  
 self.pharmacy\_table.insert("", END, values=i)  
 conn.commit()  
 else:  
 messagebox.showinfo("Invalid data", "Data not available")  
  
 conn.close()  
  
 *# def search\_data1(self):  
 # #pass  
 # conn = sqlite3.connect("mydata.db")  
 # my\_cursor = conn.cursor()  
 # my\_cursor.execute("select \* from pharmacy where " + str(  
 # self.search\_var.get() + " LIKE '" + str(self.searchTxt\_var1.get()) + "%'"))  
 # rows = my\_cursor.fetchall()  
 # if len(rows) != 0:  
 # self.pharmacy\_table.delete(\*self.pharmacy\_table.get\_children())  
 # for i in rows:  
 # self.pharmacy\_table.insert("", END, values=i)  
 # conn.commit()  
 # else:  
 # messagebox.showinfo("Invalid data", "Data not available")  
 #  
 # conn.close()* def refresh\_w(self):  
 self.destroy()  
 self.\_\_init\_\_(root)  
  
  
  
 *# def Max\_window(self):  
 # self.MaxWindow = Toplevel(self.root)  
 # self.obj = DATABASE\_PAGE(self.MaxWindow)* def LogOut\_w(self):  
 self.new\_exit\_win=Toplevel()  
 self.new\_exit\_win.focus()  
 self.new\_exit\_win.title("Apollo Pharmacy")  
 self.new\_exit\_win.geometry("300x145+1030+130")  
 *#self.new\_exit\_win.configure(bg="#ffe0e0")* self.new\_exit\_win.iconbitmap("photos/Avatar-doctor.ico")  
  
 lbltitle = Label(self.new\_exit\_win, text="Apollo Pharmacy", bd=2, relief=RIDGE  
 , bg="white", fg="darkgreen", font=("times new roman", 16, "bold"))  
 lbltitle.pack(side=TOP, fill=X)  
  
 img\_logo\_new = Image.open("photos/logo2.jpg")  
 img\_logo\_new = img\_logo\_new.resize((25, 25), Image.ANTIALIAS)  
 self.logo = ImageTk.PhotoImage(img\_logo\_new)  
 lbl\_logo = Label(self.new\_exit\_win, image=self.logo, borderwidth=0)  
 lbl\_logo.place(x=40, y=3)  
  
  
 img\_back\_home = Image.open("photos/home\_btn.png")  
 img\_back\_home = img\_back\_home.resize((70, 70), Image.ANTIALIAS)  
 self.home\_btn = ImageTk.PhotoImage(img\_back\_home)  
 bt\_back = Button(self.new\_exit\_win, image=self.home\_btn, relief=RAISED, borderwidth=3, bg="white", command=self.Back\_bt\_Main)  
 bt\_back.place(x=15, y=48)  
  
 self.lbl\_home\_msg = tk.Label(self.new\_exit\_win, text="", font=("times new roman", 8, "bold"),width=8)  
 self.lbl\_home\_msg.place(x=22,y=126)  
  
 bt\_back.bind("<Enter>", self.on\_enter)  
 bt\_back.bind("<Leave>", self.on\_leave)  
  
 img\_Database = Image.open("photos/data.jpg")  
 img\_Database = img\_Database.resize((70, 70), Image.ANTIALIAS)  
 self.data\_btn = ImageTk.PhotoImage(img\_Database)  
 bt\_Database = Button(self.new\_exit\_win, image=self.data\_btn, relief=RAISED, borderwidth=3, bg="white", command=self.Max\_Window)  
 bt\_Database.place(x=110, y=48)  
  
 self.lbl\_data\_msg = tk.Label(self.new\_exit\_win, text="", font=("times new roman", 8, "bold"),width=8)  
 self.lbl\_data\_msg.place(x=122,y=126)  
  
 bt\_Database.bind("<Enter>", self.on\_enter0)  
 bt\_Database.bind("<Leave>", self.on\_leave0)  
  
 img\_login\_back = Image.open("photos/login\_page\_btn.jpg")  
 img\_login\_back = img\_login\_back.resize((70, 70), Image.ANTIALIAS)  
 self.login\_page\_btn = ImageTk.PhotoImage(img\_login\_back)  
 bt\_login\_back = Button(self.new\_exit\_win, image=self.login\_page\_btn, relief=RAISED, borderwidth=3, bg="white", command=self.Back\_bt\_login)  
 bt\_login\_back.place(x=205, y=48)  
  
 self.lbl\_login\_msg = tk.Label(self.new\_exit\_win, text="", font=("times new roman", 8, "bold"),width=7,)  
  
 self.lbl\_login\_msg.place(x=218,y=126)  
  
 bt\_login\_back.bind("<Enter>", self.on\_enter1)  
 bt\_login\_back.bind("<Leave>", self.on\_leave1)  
  
  
 def on\_enter(self, event):  
 self.lbl\_home\_msg.configure(text="Home Page")  
  
 def on\_leave(self, enter):  
 self.lbl\_home\_msg.configure(text="")  
  
 def on\_enter0(self, event):  
 self.lbl\_data\_msg.configure(text="Database")  
  
 def on\_leave0(self, enter):  
 self.lbl\_data\_msg.configure(text="")  
  
 def on\_enter1(self, event):  
 self.lbl\_login\_msg.configure(text="Logout")  
  
 def on\_leave1(self, enter):  
 self.lbl\_login\_msg.configure(text="")  
  
  
 def Back\_bt\_Main(self):  
 root.destroy()  
 call(['python','main.py'])  
  
 *# self.new\_exit\_win.withdraw()  
 # self.Front\_page = Toplevel(self.new\_exit\_win)  
 # self.obj = Main\_page(self.Front\_page)* def Max\_Window(self):  
 root.destroy()  
 call(['python','DataBase.py'])  
 *# self.new\_exit\_win.withdraw()  
 # self.max\_page = Toplevel(self.new\_exit\_win)  
 # self.obj = PharmaLogin(self.max\_page)* def Back\_bt\_login(self):  
 root.destroy()  
 call(['python','login.py'])  
 *# self.new\_exit\_win.withdraw()  
 # self.login\_page = Toplevel(self.new\_exit\_win)  
 # self.obj = PharmaLogin(self.login\_page)*if \_\_name\_\_ == '\_\_main\_\_':  
 root = Tk()  
 obj = ApolloPharmacy(root).pack(side="top", fill="both", expand="true")  
 messagebox.showinfo("Welcome",'Welcome to Apollo Pharmacy')  
 root.mainloop()

**Output:**

**\**



**Database window**

from tkinter import \*  
from PIL import Image, ImageTk  
from tkinter import ttk  
from tkinter import messagebox  
import sqlite3  
from time import strftime  
from datetime import date  
from subprocess import call  
  
  
*# ===========================Details Window======================*class DATABASE\_PAGE:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Apollo Pharmacy")  
 self.root.geometry("1550x800+0+0")  
 self.root.iconbitmap("photos/Avatar-doctor.ico")  
 self.root.state('zoomed') *# open main\_frame\_window in maximize state.  
  
 # =========================Heading==================================* lbltitle = Label(self.root, text="Apollo Pharmacy", bd=15, relief=RIDGE  
 , bg="white", fg="darkgreen", font=("times new roman", 35, "bold"), padx=2, pady=4)  
 lbltitle.pack(side=TOP, fill=X)  
  
 *# img1 = Image.open("C:\\Users\\bhanu\\OneDrive\\Desktop\\pharma image\\Bhanu-logo.jpg")* img1 = Image.open("photos/logo2.jpg")  
 img1 = img1.resize((60, 58), Image.ANTIALIAS)  
 self.photoimg1 = ImageTk.PhotoImage(img1)  
 b1 = Button(self.root, image=self.photoimg1, borderwidth=0)  
 b1.place(x=440, y=15)  
  
 img\_back = Image.open("photos/backicon.png")  
 img\_back = img\_back.resize((20, 16), Image.ANTIALIAS)  
 self.photoimg11 = ImageTk.PhotoImage(img\_back)  
 bt\_back = Button(self.root, image=self.photoimg11, relief=RAISED, borderwidth=2, bg="white", command=self.Back)  
 bt\_back.place(x=16, y=15)  
  
 img\_login\_back = Image.open("photos/logout.jpg")  
 img\_login\_back = img\_login\_back.resize((23, 20), Image.ANTIALIAS)  
 self.photoimg12 = ImageTk.PhotoImage(img\_login\_back)  
 bt\_login\_back = Button(self.root, image=self.photoimg12, relief=RAISED, borderwidth=2, bg="white",  
 command=self.LogOut\_w)  
 bt\_login\_back.place(x=1320, y=15)  
  
 *# ===================main variable========================* self.ref\_var = StringVar()  
 self.cmpName\_var = StringVar()  
 self.typeMed\_var = StringVar()  
 self.medName\_var = StringVar()  
 self.lot\_var = StringVar()  
 self.issuedate\_var = StringVar()  
 self.expdate\_var = StringVar()  
 self.uses\_var = StringVar()  
 self.sideeffect\_var = StringVar()  
 self.warning\_var = StringVar()  
 self.dosage\_var = StringVar()  
 self.price\_var = StringVar()  
 self.product\_var = StringVar()  
  
 *# =========================DateTime=====================* def time():  
 string = strftime('%H:%M:%S %p')  
 lb1.config(text=string)  
 lb1.after(1000, time)  
  
 lb1 = Label(lbltitle, font=('times new roman', 11, 'bold'), background='white', foreground='blue')  
 lb1.place(x=0, y=20, width=120, height=40)  
 time()  
  
 today = date.today()  
 string = today.strftime("%d/%m/%Y")  
 lb1.config(text=string)  
  
 lb1 = Label(lbltitle, font=('times new roman', 11, 'bold'), background='white', foreground='blue')  
 lb1.place(x=1200, y=20, width=120, height=40)  
 time()  
  
 *# =====================Search BY=========================* DataSearchFrame = Frame(self.root, bd=12, relief=RIDGE, padx=0)  
 DataSearchFrame.place(x=0, y=93, width=1360, height=55)  
  
 lblSearch = Label(DataSearchFrame, font=("arial", 15, "bold"), text="Search By", bg="gray", fg="white")  
 lblSearch.place(x=0, y=0, width=250, height=30)  
  
 *# variable* self.search\_var = StringVar()  
 search\_combo = ttk.Combobox(DataSearchFrame, textvariable=self.search\_var, width=12, font=("arial", 14, "bold"),  
 state="readonly")  
 search\_combo["values"] = ("Ref\_no", "medName", "lotNo")  
 search\_combo.place(x=220, y=0, width=200, height=30)  
 search\_combo.current(0)  
  
 *# variable2* self.searchTxt\_var = StringVar()  
 txtSearch = Entry(DataSearchFrame, textvariable=self.searchTxt\_var, bd=3, relief=RIDGE, width=100,  
 font=("arial", 12, "bold"))  
 txtSearch.place(x=420, y=0, width=220, height=30)  
  
 searchBtn = Button(DataSearchFrame, text="SEARCH", padx=9, font=("arial", 12, "bold"), relief=RAISED, bd=5,  
 bg="darkgreen", fg="white",  
 command=self.search\_data) *# lambda:ApolloPharmacy.search\_data)* searchBtn.place(x=670, y=2, width=180, height=28)  
  
 showAll = Button(DataSearchFrame, text="SHOW ALL", padx=10, font=("arial", 12, "bold"), relief=RAISED, bd=5,  
 bg="darkgreen", fg="white",  
 command=self.fetch\_data)  
 showAll.place(x=880, y=2, width=180, height=28)  
  
 *# ==================================================* DataFrame = LabelFrame(self.root, bd=15, text="Medicine Details", font=("arial", 12, "bold"), relief=RIDGE,  
 bg="darkgreen", fg="white")  
 DataFrame.place(x=0, y=150, width=1360, height=545)  
  
 *# =============Main Table & scrollbar ===================* Table\_frame = Frame(DataFrame, bd=5, relief=SUNKEN)  
 Table\_frame.place(x=0, y=0, width=1325, height=505)  
  
 scroll\_x = ttk.Scrollbar(Table\_frame, orient=HORIZONTAL)  
 scroll\_x.pack(side=BOTTOM, fill=X)  
 scroll\_y = ttk.Scrollbar(Table\_frame, orient=VERTICAL)  
 scroll\_y.pack(side=RIGHT, fill=Y)  
  
 self.pharmacy\_table = ttk.Treeview(Table\_frame, column=("reg", "companyname", "type", "tabletname", "lotno",  
 "issuedate", "expdate", "uses", "sideeffect", "warning",  
 "dosage",  
 "price", "productqt"), xscrollcommand=scroll\_x.set,  
 yscrollcommand=scroll\_y.set)  
 scroll\_x.pack(side=BOTTOM, fill=X)  
 scroll\_y.pack(side=RIGHT, fill=Y)  
  
 scroll\_x.config(command=self.pharmacy\_table.xview)  
 scroll\_y.config(command=self.pharmacy\_table.yview)  
  
 self.pharmacy\_table["show"] = "headings"  
  
 self.pharmacy\_table.heading("reg", text="Ref No")  
 self.pharmacy\_table.heading("companyname", text="Company Name")  
 self.pharmacy\_table.heading("type", text="Medicine Type")  
 self.pharmacy\_table.heading("tabletname", text="Tablet Name")  
 self.pharmacy\_table.heading("lotno", text="Lot No")  
 self.pharmacy\_table.heading("issuedate", text="Issue Date")  
 self.pharmacy\_table.heading("expdate", text="Exp Date")  
 self.pharmacy\_table.heading("uses", text="Uses")  
 self.pharmacy\_table.heading("sideeffect", text="Side Effect")  
 self.pharmacy\_table.heading("warning", text="Prec&Warning")  
 self.pharmacy\_table.heading("dosage", text="Dosage")  
 self.pharmacy\_table.heading("price", text="Price")  
 self.pharmacy\_table.heading("productqt", text="Quantity")  
 self.pharmacy\_table.pack(fill=BOTH, expand=1)  
  
 self.pharmacy\_table.column("reg", width=40)  
 self.pharmacy\_table.column("companyname", width=100)  
 self.pharmacy\_table.column("type", width=90)  
 self.pharmacy\_table.column("tabletname", width=80)  
 self.pharmacy\_table.column("lotno", width=50)  
 self.pharmacy\_table.column("issuedate", width=70)  
 self.pharmacy\_table.column("expdate", width=70)  
 self.pharmacy\_table.column("uses", width=150)  
 self.pharmacy\_table.column("sideeffect", width=150)  
 self.pharmacy\_table.column("warning", width=130)  
 self.pharmacy\_table.column("dosage", width=110)  
 self.pharmacy\_table.column("price", width=40)  
 self.pharmacy\_table.column("productqt", width=50)  
 self.fetch\_data()  
  
 *# ========================================================* def fetch\_data(self):  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("select \* from pharmacy")  
 row = my\_cursor.fetchall()  
 if len(row) != 0:  
 self.pharmacy\_table.delete(\*self.pharmacy\_table.get\_children())  
 for i in row:  
 self.pharmacy\_table.insert("", END, values=i)  
 conn.commit()  
 conn.close()  
  
 def search\_data(self):  
 conn = sqlite3.connect("mydata.db")  
 my\_cursor = conn.cursor()  
 my\_cursor.execute("select \* from pharmacy where " + str(  
 self.search\_var.get() + " LIKE '" + str(self.searchTxt\_var.get()) + "%'"))  
 rows = my\_cursor.fetchall()  
 if len(rows) != 0:  
 self.pharmacy\_table.delete(\*self.pharmacy\_table.get\_children())  
 for i in rows:  
 self.pharmacy\_table.insert("", END, values=i)  
 conn.commit()  
 else:  
 messagebox.showinfo("Invalid data", "Data not available")  
 conn.close()  
  
 def LogOut\_w(self):  
 self.new\_exit\_win = Toplevel()  
 self.new\_exit\_win.focus()  
 self.new\_exit\_win.title("Apollo Pharmacy")  
 self.new\_exit\_win.geometry("200x135+1110+130")  
 self.new\_exit\_win.configure(bg="#ffe0e0")  
 self.new\_exit\_win.iconbitmap("photos/Avatar-doctor.ico")  
  
  
  
 lbltitle = Label(self.new\_exit\_win, text="Apollo Pharmacy", bd=2, relief=RIDGE  
 , bg="white", fg="darkgreen", font=("times new roman", 10, "bold"))  
 lbltitle.pack(side=TOP, fill=X)  
  
 img\_logo\_new = Image.open("photos/logo2.jpg")  
 img\_logo\_new = img\_logo\_new.resize((15, 15), Image.ANTIALIAS)  
 self.logo = ImageTk.PhotoImage(img\_logo\_new)  
 lbl\_logo = Label(self.new\_exit\_win, image=self.logo, borderwidth=0)  
 lbl\_logo.place(x=35, y=3)  
  
 img\_back = Image.open("photos/home\_btn.png")  
 img\_back = img\_back.resize((70, 70), Image.ANTIALIAS)  
 self.home\_btn = ImageTk.PhotoImage(img\_back)  
 bt\_back = Button(self.new\_exit\_win, image=self.home\_btn, relief=RAISED, borderwidth=3, bg="white",  
 command=self.Back\_bt\_Main)  
 bt\_back.place(x=15, y=38)  
  
 self.lbl\_home\_msg = Label(self.new\_exit\_win, text="", font=("times new roman", 8, "bold"), bg="#ffe0e0",width=8)  
 self.lbl\_home\_msg.place(x=22, y=117)  
  
 bt\_back.bind("<Enter>", self.on\_enter)  
 bt\_back.bind("<Leave>", self.on\_leave)  
  
 img\_login\_back = Image.open("photos/login\_page\_btn.jpg")  
 img\_login\_back = img\_login\_back.resize((70, 70), Image.ANTIALIAS)  
 self.login\_page\_btn = ImageTk.PhotoImage(img\_login\_back)  
 bt\_login\_back = Button(self.new\_exit\_win, image=self.login\_page\_btn, relief=RAISED, borderwidth=3, bg="white",  
 command=self.Back\_bt\_login)  
 bt\_login\_back.place(x=110, y=38)  
  
 self.lbl\_login\_msg = Label(self.new\_exit\_win, text="", font=("times new roman", 8, "bold"), bg="#ffe0e0", width=8)  
  
 self.lbl\_login\_msg.place(x=120, y=117)  
  
 bt\_login\_back.bind("<Enter>", self.on\_enter1)  
 bt\_login\_back.bind("<Leave>", self.on\_leave1)  
 *# ===========================================================* def on\_enter(self, event):  
 self.lbl\_home\_msg.configure(text="Home Page")  
  
 def on\_leave(self, enter):  
 self.lbl\_home\_msg.configure(text="")  
  
 def on\_enter1(self, event):  
 self.lbl\_login\_msg.configure(text="Logout")  
  
 def on\_leave1(self, enter):  
 self.lbl\_login\_msg.configure(text="")  
  
 *# ============================================================* def Back\_bt\_Main(self):  
 root.destroy()  
 call(['python', 'main.py'])  
  
 *# self.new\_exit\_win.withdraw()  
 # self.Front\_page = Toplevel(self.new\_exit\_win)  
 # self.obj = Main\_page(self.Front\_page)* def Back\_bt\_login(self):  
 root.destroy()  
 call(['python', 'login.py'])  
 *# self.new\_exit\_win.withdraw()  
 # self.login\_page = Toplevel(self.new\_exit\_win)  
 # self.obj = PharmaLogin(self.login\_page)* def Back(self):  
 root.destroy()  
 call(['python', 'apollo\_main\_page.py'])  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 root = Tk()  
 obj = DATABASE\_PAGE(root)  
 root.mainloop()

**Output:**

