

MEDICAL STORE MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

RAKSHIT AGARWAL(RA2011026010340)
TAPNANSHU ATHARVA(RA2011026010309)

Under the guidance of

Dr. S. Sadagopan
(Assistant Professor, Dept Of Computational
Intelligence)

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING



SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

APRIL 2023



COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this project report “ **Medical Store Management System**” is the bonafide work of “ **Rakshit Agarwal(RA2011026010340) , Tapnanshu Atharva(RA2011026010309)** “, of III Year/VI Sem B.tech(CSE) who carried out the mini project work under my supervision for the course 18CSC303J- Database Management systems in SRM Institute of Science and Technology during the academic year 2022-2023(Even semester).

SIGNATURE

Dr. R. Annie Uthra

Professor and Head

Department of computational intelligence

SIGNATURE

Dr. S. Sadagopan

Assistant Professor

ABSTRACT

The Medical Store Management System is a software application that helps manage the operations of a medical store or pharmacy. The system provides a platform for managing inventory, sales, purchases, and other activities related to the store's operations. The system aims to improve the efficiency and accuracy of store management by automating many of the manual processes involved. The project involves the design and implementation of a user-friendly interface that allows users to manage the store's inventory and sales. The system enables users to add and remove products, track inventory levels, set prices, and view sales reports. The system also includes features for managing customers, suppliers, and employee records. The project will be implemented using the latest software development methodologies and techniques, including agile development, object-oriented programming, and database design. The system will be designed to be scalable and flexible, allowing for future expansion and customization as needed. One of the key benefits of the Medical Store Management System is that it streamlines the inventory management process. The system enables users to track inventory levels in real-time, ensuring that the store always has sufficient stock to meet customer demand. The system also provides tools for analyzing sales data and forecasting future demand, enabling users to make informed decisions about inventory management. Another benefit of the system is that it simplifies the sales process. The system provides an easy-to-use interface for creating and processing sales orders, reducing the time and effort required to complete each transaction. The system also enables users to generate invoices and receipts automatically, reducing the risk of errors and improving the accuracy of financial records. Overall, the Medical Store Management System is a valuable tool for managing the operations of a medical store or pharmacy. It improves efficiency, accuracy, and customer satisfaction, enabling store owners to focus on providing high-quality products and services.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	ABSTRACT	iii
	TABLE OF CONTENTS	iv
	LIST OF FIGURES	v
	LIST OF TABLES	vi
	ABBREVIATIONS	vii
1	INTRODUCTION	8
1.1	Introduction	8
1.2	Problem statement	9
1.3	Objectives	10
1.4	Scope and applications	11
1.5	General and Unique Services in the database	12
1.6	Software Requirements Specification	12
2	LITERATURE SURVEY	13
2.1	Existing system	13
2.2	Comparison of Existing vs Proposed system	14
3	SYSTEM ARCHITECTURE AND DESIGN	15
3.1	Architecture Diagram	15
3.1.1	Front end (UI) design	16
3.1.2	Back end (Database) design	18
3.2	ER Diagram and Use case Diagram	23
4	Modules and Functionalities	24
5	CODING AND TESTING	25
6	RESULTS AND DISCUSSIONS	34
7	CONCLUSION AND FUTURE ENHANCEMENT	35
8	REFERENCES	36

LIST OF FIGURES

Figure No.	Figure Name	Page No.
4.1	ER DIAGRAM	23
4.2	ARCHITECTURE DIAGRAM	15
4.3	USECASE DIAGRAM	23
4.4	FRONTEND DESIGN	16
5.1	BACKEND DESIGN	18
5.2	TESTING	31

LIST OF TABLES

Table No.	Table Name	Page No.
2.1	General and Unique Services	12
2.2	Software Requirements	12
3.1	Modules and Functionalities	24

ABBREVIATIONS

AES	Advanced Encryption Standard
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
CV	Computer Vision
DB	Data Base
SQL	Structured Query Language
UI	User Interface

CHAPTER1

INTRODUCTION

1.1 Introduction of Medical Store System Project

The Medical Store Management System is an essential software application that is designed to automate and simplify the management of a medical store or pharmacy. The system provides an efficient platform for managing the inventory, sales, purchases, and other essential activities related to the store's operations. The system aims to improve the accuracy and efficiency of the store management process by eliminating the manual methods that are often prone to human error.

The traditional methods of managing a medical store involve a lot of paperwork, and it can be quite challenging to manage inventory and sales records manually. This can lead to confusion, errors, and inconsistencies, which can have a significant impact on the store's performance. The Medical Store Management System is designed to address these challenges and make store management much more manageable.

The system provides a user-friendly interface that simplifies the process of managing inventory and sales records. The system enables users to add and remove products, track inventory levels, set prices, and view sales reports, all from a single platform. The system also provides tools for managing customer, supplier, and employee records, making it easier to keep track of all aspects of the store's operations.

The Medical Store Management System is designed to be scalable and flexible, enabling store owners to adapt to changes in the market and customer demand. The system provides tools for analyzing sales data and forecasting future demand, enabling users to make informed decisions about inventory management. This ensures that the store always has sufficient stock to meet customer demand, reducing the risk of stockouts and lost sales.

In summary, the Medical Store Management System is a valuable tool for managing the operations of a medical store or pharmacy. It provides an efficient platform for managing inventory, sales, and other essential activities, reducing the time and effort required to complete each task. The system improves the accuracy and efficiency of the store management process, enabling store owners to focus on providing high-quality products and services to their customers.

1.2 Problem Statement

The problem that the Medical Store Management System aims to address is the inefficiency and errors associated with the manual methods of managing a medical store or pharmacy. The traditional methods involve a lot of paperwork, and it can be challenging to keep track of inventory, sales, and other essential activities manually. This can lead to confusion, errors, and inconsistencies in the store's operations, which can impact its performance negatively.

Moreover, managing inventory can be a daunting task, especially for a medical store that deals with a vast range of products. It is challenging to track the stock levels of different products and ensure that the store always has sufficient stock to meet customer demand. Inaccurate inventory management can result in stockouts, leading to lost sales and dissatisfied customers.

The manual methods of managing sales and customer records can also be time-consuming and prone to errors. Creating sales orders, processing transactions, and generating invoices and receipts manually can take up a lot of time, and it is easy to make mistakes that can impact the accuracy of financial records.

Furthermore, managing employee and supplier records manually can also be challenging, especially for stores with a large number of employees and suppliers. Keeping track of employee attendance, payroll, and other essential details can be quite tedious and time-consuming.

1.3 Objectives

The primary objective of the Medical Store Management System is to automate and simplify the management of a medical store or pharmacy. The system aims to provide an efficient platform for managing inventory, sales, purchases, and other essential activities related to the store's operations. The following are some of the specific objectives of the system:

Inventory Management: The system aims to provide an efficient tool for managing inventory. The system enables users to add and remove products, track inventory levels, set prices, and view sales reports, all from a single platform. The system also provides tools for analyzing sales data and forecasting future demand, enabling users to make informed decisions about inventory management.

Sales Management: The system aims to automate and simplify the process of managing sales records. The system provides tools for creating sales orders, processing transactions, generating invoices and receipts, and managing customer records. The system also enables users to track sales performance, view sales reports, and analyze sales data to make informed decisions.

1.4 Scope & Application

The scope of the Medical Store Management System is extensive and can be applied to various types of medical stores or pharmacies. The system's primary application is to manage the store's inventory, sales, purchases, and other essential operations related to the store's management. The following are some of the significant areas of application of the Medical Store Management System:

Inventory Management: The system can manage a wide range of inventory items, including medicines, medical devices, and other healthcare products. The system can handle inventory management for single stores or multiple branches of a chain of stores.

Sales Management: The system can handle sales records, including creating sales orders, processing transactions, generating invoices and receipts, and managing customer records. The system can also provide real-time sales reports and analytics.

Purchase Management: The system can handle the store's purchase records, including creating purchase orders, managing supplier records, and tracking purchase orders' status.

Employee Management: The system can manage employee records, including attendance, payroll, and other essential details related to the store's workforce.

Reporting and Analytics: The system can provide a variety of reports and analytics related to inventory, sales, purchases, and other essential metrics. The system can also provide real-time data analysis and forecasting tools to help store managers make informed decisions about the store's operations.

1.5 General & unique services of a database

- Menu Driven
- Paperless Practice
- Improve efficiency
- cost efficient
- importing of drug list
- maintain customer relationship
- provide multi user environment

1.6 Software Requirements

- MySQL
- Python
- StreamLit
- SQLite
- HTML , CSS
- Processor : 4GB & Above
- Memory – 6GB RAM
- Hard Disk – 10 GB HDD
- Internet browser – cross platform

2 Literature Survey

2.1 Existing system

The existing medical store management system relied heavily on manual processes, which were time-consuming and prone to errors. Store managers had to manually track inventory levels, manage sales and purchases, and keep track of employee records, among other things. This system was not only inefficient but also made it difficult to track essential data, such as sales performance, inventory levels, and employee productivity.

One of the main drawbacks of the existing system was the lack of real-time data. Without access to up-to-date information, store managers had to rely on guesswork when making decisions about inventory management, sales, and other essential metrics. This often resulted in overstocking or understocking of inventory, leading to either lost sales or waste.

Another issue with the existing system was the lack of automation. The manual processes not only took up valuable time but also increased the risk of errors, such as incorrect inventory counts or inaccurate sales records. These errors could lead to significant financial losses and affect customer satisfaction.

Overall, the existing medical store management system was inefficient and ineffective, requiring significant time and effort to manage. Store managers needed to find a more efficient and effective way to manage their operations, which led to the development of modern medical store management systems.

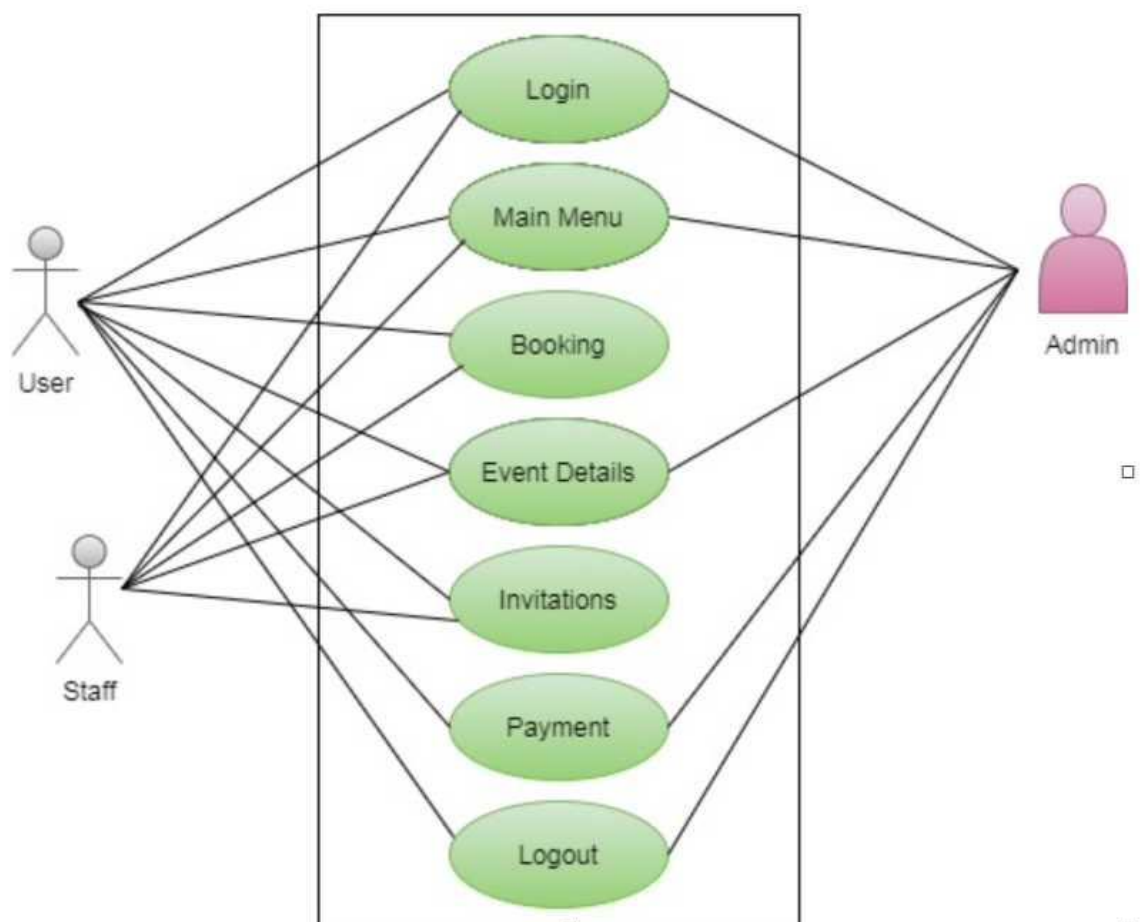
2.2 Existing system VS New System

There are several key differences between the old and modern medical store management systems:

1. **Automation:** The modern medical store management system is highly automated, with many processes automated, such as inventory management, sales management, purchase management, and employee management. This automation significantly reduces the amount of manual effort required to manage the store and reduces the risk of errors.
2. **Real-time Data:** Unlike the old system, modern medical store management systems provide real-time data on inventory levels, sales, purchases, employee performance, and other critical metrics.
3. **Scalability:** Modern medical store management systems are designed to scale as the business grows, making it easy to add new products, manage multiple locations, and expand the business.
4. **Integration with other systems:** Modern medical store management systems can integrate with other systems, such as electronic medical records (EMR) and billing systems, enabling seamless information exchange between different departments and systems.
5. **Analytics:** Modern medical store management systems offer advanced analytics and reporting features, allowing store managers to track sales performance, inventory levels, employee productivity, and other critical metrics.

3 System Architect And Design

3.1 Architecture Diagram



3.1.1 Front-End Design

×

Menu

Admin

User Name

admin

Password

•••••

Menu

Drugs

Menu

Add

☰

Pharmacy Database Dashboard

Add Drugs

Enter the Drug Name

Enter the quantity

Expiry Date of Drug (YYYY-MM-DD)

2023/04/28

Enter the Drug id (example:#D1)

When to Use

Add Drug

Made with Streamlit



Menu

SignUp



Create New Account

Name

Password



Confirm Password

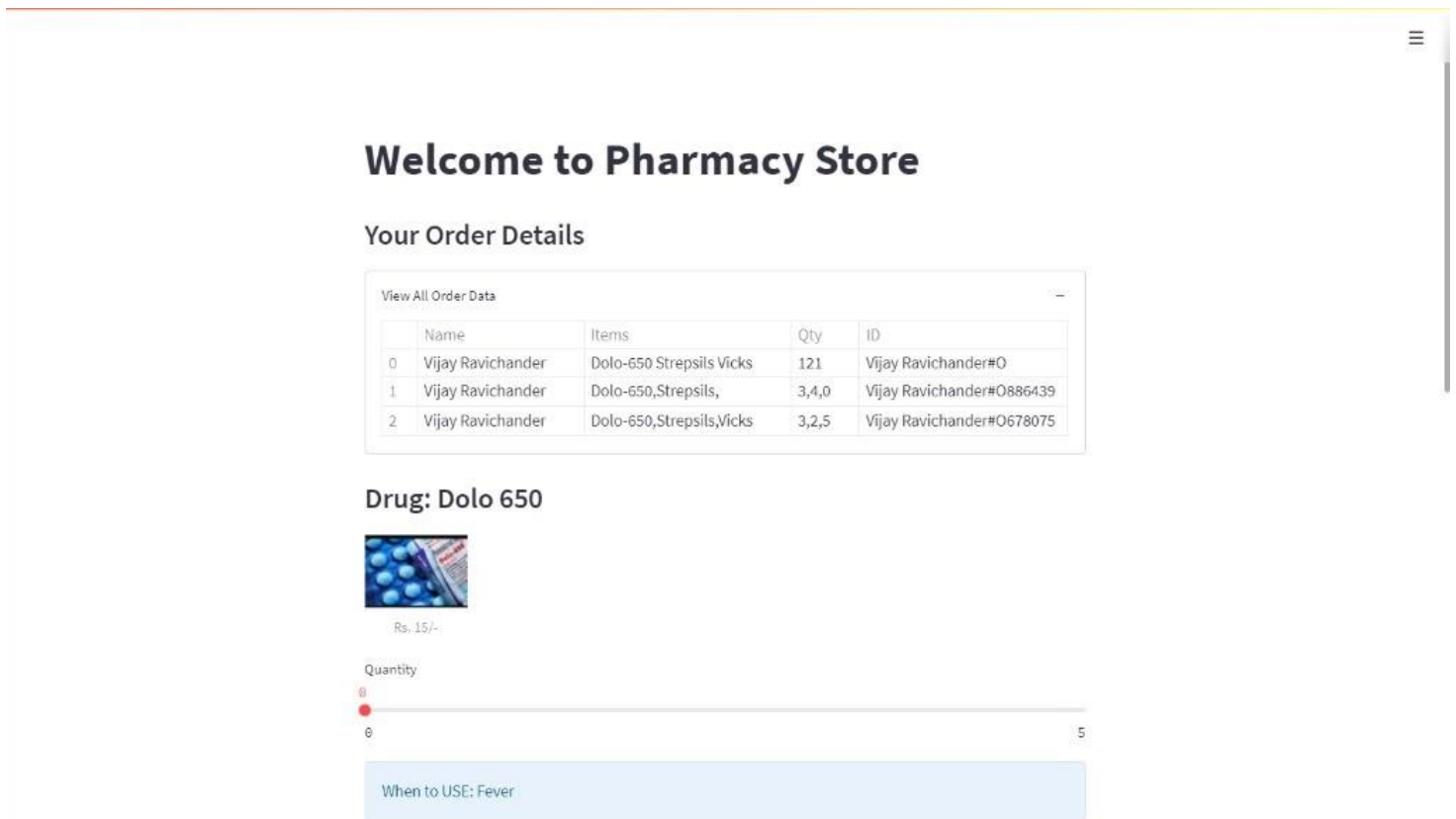
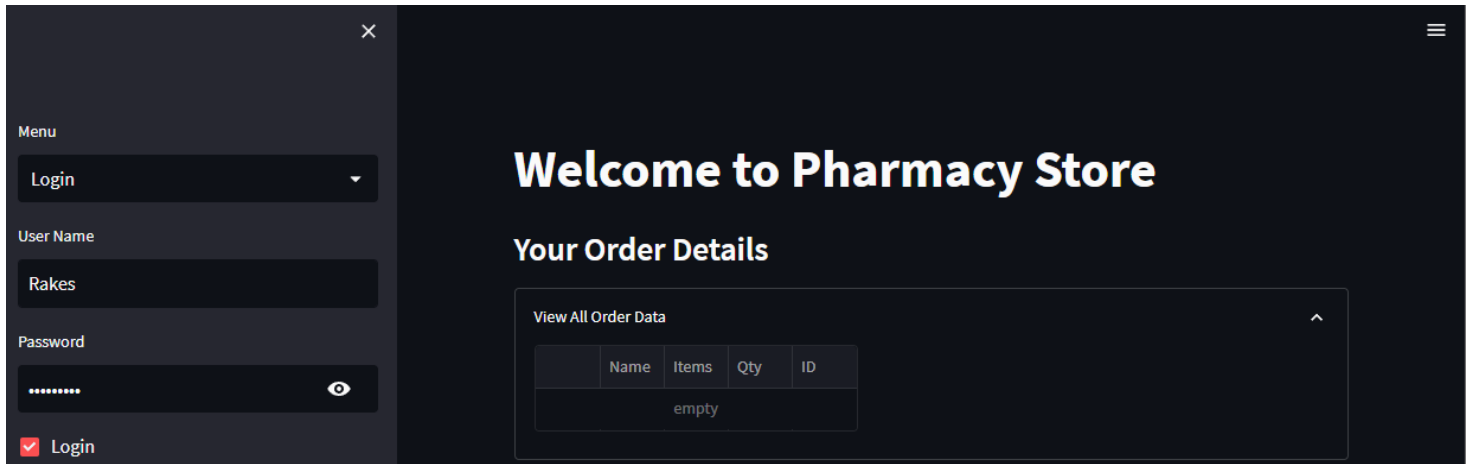


Email ID

State

Phone Number

Signup



3.1.2 Back End Design

```
C:\Users\hatka> OneDrive\Desktop> dms> Pharmacy_Management_System-main> Pharmacy_Management_System-main> drugdatabase.sql
1 CREATE SCHEMA drugdatabase;
2
3 USE drugdatabase;
4
5 CREATE TABLE customer (
6     uid varchar(20) NOT NULL,
7     pass varchar(20) DEFAULT NULL,
8     fname varchar(15) DEFAULT NULL,
9     lname varchar(15) DEFAULT NULL,
10    email varchar(30) DEFAULT NULL,
11    address varchar(128) DEFAULT NULL,
12    phno bigint DEFAULT NULL,
13    PRIMARY KEY (uid)
14 );
15
16 CREATE TABLE seller (
17     sid varchar(15) NOT NULL,
18     sname varchar(20) DEFAULT NULL,
19     pass varchar(20) DEFAULT NULL,
20     address varchar(128) DEFAULT NULL,
21     phno bigint DEFAULT NULL,
22     PRIMARY KEY (sid)
23 );
24
25 CREATE TABLE product (
26     pid varchar(15) NOT NULL,
27     pname varchar(20) DEFAULT NULL,
28     manufacturer varchar(20) DEFAULT NULL,
29     mfg date DEFAULT NULL,
30     exp date DEFAULT NULL,
31     price int DEFAULT NULL,
32     PRIMARY KEY (pid),
33     UNIQUE KEY pname (pname)
34 );
35
36 CREATE TABLE inventory (
37     pid varchar(15) NOT NULL,
38     pname varchar(20) DEFAULT NULL,
39     quantity int unsigned DEFAULT NULL,
40     sid varchar(15) NOT NULL,
41     PRIMARY KEY (pid,sid),
42     CONSTRAINT fk01 FOREIGN KEY (pid) REFERENCES product (pid) ON DELETE CASCADE,
43     CONSTRAINT fk02 FOREIGN KEY (pname) REFERENCES product (pname) ON DELETE CASCADE,
44     CONSTRAINT fk03 FOREIGN KEY (sid) REFERENCES seller (sid) ON DELETE CASCADE
45 );
46
47 CREATE TABLE orders (
48     oid int NOT NULL AUTO_INCREMENT,
49     pid varchar(15) DEFAULT NULL,
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\hatka> python -u "c:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py"

DRUG Table create Successfully

Customer Table create Successfully

2023-04-27 23:41:39.917

Warning: to view this Streamlit app on a browser, run it with the following command:

streamlit run c:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py [ARGUMENTS]

PS C:\Users\hatka> streamlit run c:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501

Network URL: http://192.168.1.2:8501

DRUG Table create Successfully

Customer Table create Successfully

DRUG Table create Successfully

Customer Table create Successfully

DRUG Table create Successfully

Customer Table create Successfully

DRUG Table create Successfully

Customer Table create Successfully

DRUG Table create Successfully

Customer Table create Successfully

2023-04-27 23:42:47.870 Uncaught app exception

Traceback (most recent call last):

File "C:\Users\hatka\AppData\Local\Programs\Python\Python311\Lib\site-packages\streamlit\runtime\script_runner.py", line 565, in _run_script

exec(code, module.__dict__)

File "C:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py", line 292, in <module>

customer(username, password)

File "C:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py", line 223, in customer

if getauthenticate(username, password):

^^

File "C:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py", line 212, in getauthenticate

if cust_password[0][0] == password:

^^^^^^^^^^^^^^^^^^^^^^^^^^^^

IndexError: list index out of range

DRUG Table create Successfully

Customer Table create Successfully

DRUG Table create Successfully

Customer Table create Successfully

DRUG Table create Successfully

Customer Table create Successfully

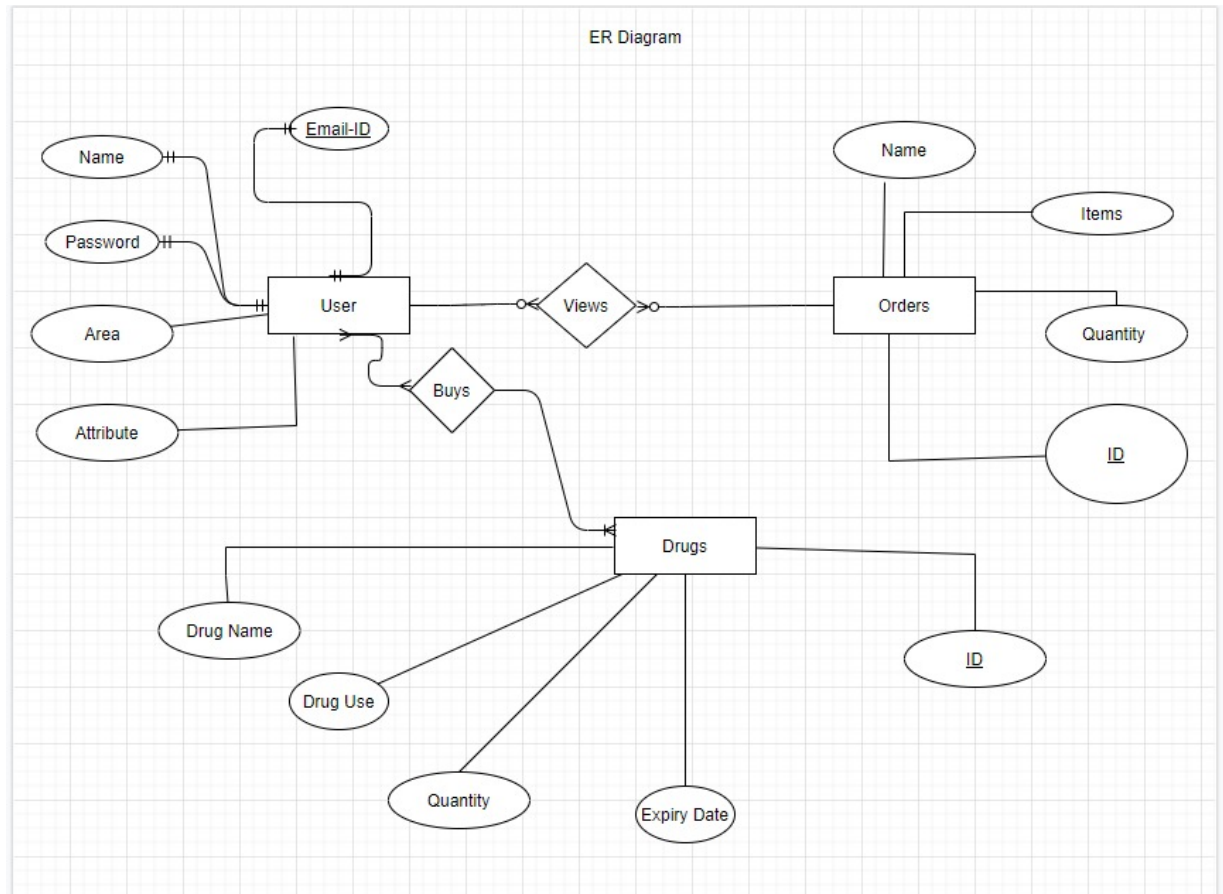
DRUG Table create Successfully

Customer Table create Successfully

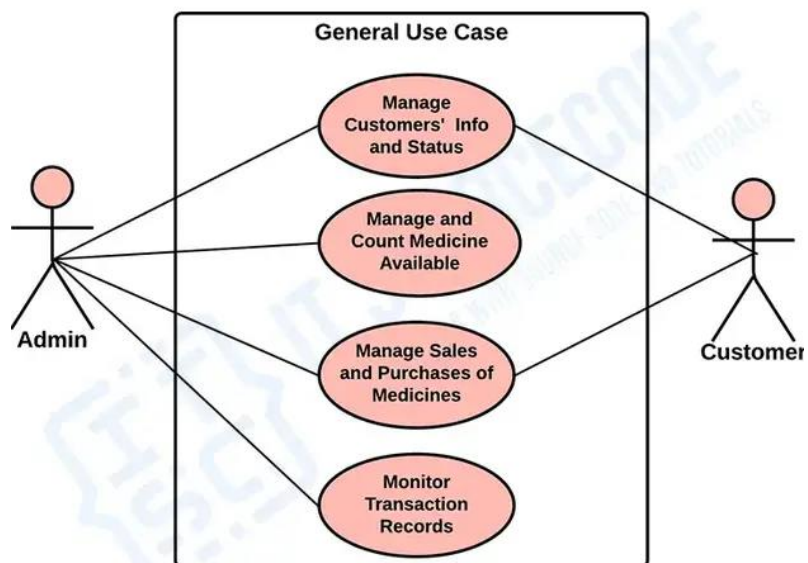
DRUG Table create Successfully

Customer Table create Successfully

3.2 ER & USE CASE DIAGRAM



PHARMACY MANAGEMENT SYSTEM



USE CASE DIAGRAM

4 Modules and Functionalities

A medical store management system typically includes the following modules and functionalities:

1. **Inventory management:** This module allows the user to track the inventory levels of drugs, medical equipment, and other products in the store, and manage the stock levels by adding or removing items from the inventory. It may also include features such as automated reordering and alerts for low stock levels.
2. **Sales and billing:** This module handles the sales transactions and generates bills for customers. It includes features such as barcode scanning, discounts, and tax calculations.
3. **Customer management:** This module allows the user to manage customer information such as contact details, purchase history, and billing information.
4. **Supplier management:** This module manages the information of the suppliers who provide drugs, medical equipment, and other products to the store. It includes features such as supplier contact information, purchase orders, and payment records.
5. **Reporting and analytics:** This module generates reports and provides insights into the store's sales, inventory, and financial performance. It includes features such as sales reports, inventory reports, and profit and loss statements.
6. **Prescription management:** This module manages prescription details, including the name of the patient, medication prescribed, dosage, and the physician's name. It may also include features such as alerts for expired prescriptions and reminders for refills.
7. **Security and user management:** This module controls user access to the system and ensures data security. It includes features such as user authentication, role-based access control, and audit trails.
8. **Integration with other systems:** This module allows the medical store management system to integrate with other systems such as electronic health records (EHRs) and pharmacy benefit managers (PBMs).

5 CODING AND TESTING

5.1 CODE

```
6   import streamlit as st
7   import pandas as pd
8   from PIL import Image
9   #from drug_db import *
10  import random
11
12  ## SQL DATABASE CODE
13  import sqlite3
14
15  conn =
sqlite3.connect("drug_data.db",check_same_thread=False)
16  c = conn.cursor()
17
18  def cust_create_table():
19      c.execute("CREATE TABLE IF NOT EXISTS
Customers(
20          C_Name VARCHAR(50) NOT NULL,
21          C_Password VARCHAR(50) NOT
NULL,
22          C_Email VARCHAR(50) PRIMARY
KEY NOT NULL,
23          C_State VARCHAR(50) NOT NULL,
24          C_Number VARCHAR(50) NOT
NULL
25          )")
26      print('Customer Table create Successfully')
27
28  def customer_add_data(Cname,Cpass, Cemail,
Cstate,Cnumber):
29      c.execute("INSERT INTO Customers
(C_Name,C_Password,C_Email, C_State, C_Number)
VALUES(?,?,?,?)" , (Cname,Cpass, Cemail,
Cstate,Cnumber))
30      conn.commit()
```



```

31
32 def customer_view_all_data():
33     c.execute('SELECT * FROM Customers')
34     customer_data = c.fetchall()
35     return customer_data
36 def customer_update(Cemail,Cnumber):
37     c.execute(""" UPDATE Customers SET C_Number
= ? WHERE C_Email = ?""", (Cnumber,Cemail,))
38     conn.commit()
39     print("Updating")
40 def customer_delete(Cemail):
41     c.execute(""" DELETE FROM Customers WHERE
C_Email = ?""", (Cemail,))
42     conn.commit()
43
44 def drug_update(Duse, Did):
45     c.execute(""" UPDATE Drugs SET D_Use = ?
WHERE D_id = ?""", (Duse,Did))
46     conn.commit()
47 def drug_delete(Did):
48     c.execute(""" DELETE FROM Drugs WHERE
D_id = ?""", (Did,))
49     conn.commit()
50
51 def drug_create_table():
52     c.execute("""CREATE TABLE IF NOT EXISTS
Drugs(
53         D_Name VARCHAR(50) NOT NULL,
54         D_ExpDate DATE NOT NULL,
55         D_Use VARCHAR(50) NOT NULL,
56         D_Qty INT NOT NULL,
57         D_id INT PRIMARY KEY NOT NULL)
58     """)
59     print('DRUG Table create Successfully')
60
61 def drug_add_data(Dname, Dexpdate, Duse, Dqty,
Did):
62     c.execute("""INSERT INTO Drugs (D_Name,
D_Expdate, D_Use, D_Qty, D_id) VALUES(?,?,?,?,"""),

```

```

(Dname, Dexptime, Duse, Dqty, Did))
63     conn.commit()
64
65     def drug_view_all_data():
66         c.execute('SELECT * FROM Drugs')
67         drug_data = c.fetchall()
68         return drug_data
69
70     def order_create_table():
71         c.execute("""
72             CREATE TABLE IF NOT EXISTS Orders(
73                 O_Name VARCHAR(100) NOT NULL,
74                 O_Items VARCHAR(100) NOT NULL,
75                 O_Qty VARCHAR(100) NOT NULL,
76                 O_id VARCHAR(100) PRIMARY KEY
77             NOT NULL)
78         """)
79     def order_delete(Oid):
80         c.execute("DELETE FROM Orders WHERE
81 O_id = ?", (Oid,))
82     def
83     order_add_data(O_Name,O_Items,O_Qty,O_id):
84         c.execute("INSERT INTO Orders (O_Name,
85 O_Items,O_Qty, O_id) VALUES(?,?,?,?)",
86         (O_Name,O_Items,O_Qty,O_id))
87         conn.commit()
88
89     def order_view_data(customername):
90         c.execute('SELECT * FROM ORDERS Where
91 O_Name == ?',(customername,))
92         order_data = c.fetchall()
93         return order_data
94
95     def order_view_all_data():
96         c.execute('SELECT * FROM ORDERS')
97         order_all_data = c.fetchall()
98         return order_all_data

```

96

97

98

99 def admin():

100

101 st.title("Pharmacy Database Dashboard")

102 menu = ["Drugs", "Customers", "Orders",
"About"]

103 choice = st.sidebar.selectbox("Menu", menu)

104

105 ## DRUGS

106 if choice == "Drugs":

107

108 menu = ["Add", "View", "Update", "Delete"]

109 choice = st.sidebar.selectbox("Menu", menu)

110 if choice == "Add":

111

112 st.subheader("Add Drugs")

113

114 col1, col2 = st.columns(2)

115

116 with col1:

117 drug_name = st.text_area("Enter the Drug
Name")

118 drug_expiry = st.date_input("Expiry Date
of Drug (YYYY-MM-DD)")

119 drug_mainuse = st.text_area("When to

```

Use")
120         with col2:
121             drug_quantity = st.text_area("Enter the
quantity")
122             drug_id = st.text_area("Enter the Drug id
(example:#D1)")
123
124         if st.button("Add Drug"):
125
126             drug_add_data(drug_name,drug_expiry,drug_mainuse,dr
ug_quantity,drug_id)
127             st.success("Successfully Added Data")
128         if choice == "View":
129             st.subheader("Drug Details")
130             drug_result = drug_view_all_data()
131             #st.write(drug_result)
132             with st.expander("View All Drug Data"):
133                 drug_clean_df =
pd.DataFrame(drug_result, columns=["Name", "Expiry
Date", "Use", "Quantity", "ID"])
134                 st.dataframe(drug_clean_df)
135             with st.expander("View Drug Quantity"):
136                 drug_name_quantity_df =
drug_clean_df[['Name','Quantity']]
137                 #drug_name_quantity_df =
drug_name_quantity_df.reset_index()
138                 st.dataframe(drug_name_quantity_df)
139             if choice == 'Update':
140                 st.subheader("Update Drug Details")
141                 d_id = st.text_area("Drug ID")
142                 d_use = st.text_area("Drug Use")
143                 if st.button(label='Update'):
144                     drug_update(d_use,d_id)
145
146             if choice == 'Delete':
147                 st.subheader("Delete Drugs")
148                 did = st.text_area("Drug ID")
149                 if st.button(label="Delete"):
150                     drug_delete(did)

```

150

```
151     ## CUSTOMERS
152     elif choice == "Customers":
153
154         menu = ["View", "Update", "Delete"]
155         choice = st.sidebar.selectbox("Menu", menu)
156         if choice == "View":
157             st.subheader("Customer Details")
158             cust_result = customer_view_all_data()
159             #st.write(cust_result)
160             with st.expander("View All Customer
Data"):
161                 cust_clean_df =
pd.DataFrame(cust_result, columns=["Name",
"Password","Email-ID" ,"Area", "Number"])
162                 st.dataframe(cust_clean_df)
163
164         if choice == 'Update':
165             st.subheader("Update Customer Details")
166             cust_email = st.text_area("Email")
167             cust_number = st.text_area("Phone
Number")
168             if st.button(label='Update'):
169                 customer_update(cust_email,cust_number)
170
171         if choice == 'Delete':
172             st.subheader("Delete Customer")
173             cust_email = st.text_area("Email")
174             if st.button(label="Delete"):
175                 customer_delete(cust_email)
176
177     elif choice == "Orders":
178
179         menu = ["View"]
180         choice = st.sidebar.selectbox("Menu", menu)
181         if choice == "View":
182             st.subheader("Order Details")
```

```

183         order_result = order_view_all_data()
184         #st.write(cust_result)
185         with st.expander("View All Order Data"):
186             order_clean_df =
pd.DataFrame(order_result, columns=["Name",
"Items", "Qty" , "ID"])
187             st.dataframe(order_clean_df)
188     elif choice == "About":
189         st.subheader("DBMS Mini Project")
190         st.subheader("By Aneesh H(636) , Shruti(631) ,
Rohan(658)")
191
192 def getauthenticate(username, password):
193     #print("Auth")
194     c.execute('SELECT C_Password FROM
Customers WHERE C_Name = ?', (username,))
195     cust_password = c.fetchall()
196     #print(cust_password[0][0], "Outside password")
197     #print(password, "Parameter password")
198     if cust_password[0][0] == password:
199         #print("Inside password")
200         return True
201     else:
202         return False
203
204
#####
#####
205
206 def customer(username, password):
207     if getauthenticate(username, password):
208         print("In Customer")
209         st.title("Welcome to Pharmacy Store")
210
211         st.subheader("Your Order Details")
212         order_result = order_view_data(username)
213         # st.write(cust_result)
214         with st.expander("View All Order Data"):
215             order_clean_df =

```

```

pd.DataFrame(order_result, columns=["Name", "Items",
"Qty", "ID"])
216         st.dataframe(order_clean_df)
217
218         drug_result = drug_view_all_data()
219         print(drug_result)
220
221         st.subheader("Drug: "+drug_result[0][0])
222         img = Image.open('images/dolo650.jpg')
223         st.image(img, width=100, caption="Rs. 15/-")
224         dolo650 =
st.slider(label="Quantity",min_value=0, max_value=5,
key= 1)
225         st.info("When to USE: " +
str(drug_result[0][2]))
226
227         st.subheader("Drug: " + drug_result[1][0])
228         img = Image.open('images/strepsils.JPG')
229         st.image(img, width=100 , caption="Rs. 10/-")
230         strepsils =
st.slider(label="Quantity",min_value=0, max_value=5,
key= 2)
231         st.info("When to USE: " +
str(drug_result[1][2]))
232
233         st.subheader("Drug: " + drug_result[2][0])
234         img = Image.open('images/vicks.JPG')
235         st.image(img, width=100, caption="Rs. 65/-")
236         vicks =
st.slider(label="Quantity",min_value=0, max_value=5,
key=3)
237         st.info("When to USE: " +
str(drug_result[2][2]))
238
239         if st.button(label="Buy now"):
240             O_items = ""
241
242             if int(dolo650) > 0:
243                 O_items += "Dolo-650,"

```

```

244         if int(strepsils) > 0:
245             O_items += "Strepsils,"
246         if int(vicks) > 0:
247             O_items += "Vicks"
248         O_Qty = str(dolo650)+str(',') + str(strepsils)
+ str(",") + str(vicks)
249
250         O_id = username + "#O" +
str(random.randint(0,1000000))
251         #order_add_data(O_Name, O_Items,O_Qty,
O_id):
252         order_add_data(username, O_items, O_Qty,
O_id)
253

```

```

254 if __name__ == '__main__':
255     drug_create_table()
256     cust_create_table()
257     order_create_table()
258
259     menu = ["Login", "SignUp","Admin"]
260     choice = st.sidebar.selectbox("Menu", menu)
261     if choice == "Login":
262         username = st.sidebar.text_input("User Name")
263         password = st.sidebar.text_input("Password",
type='password')
264         if st.sidebar.checkbox(label="Login"):
265             customer(username, password)
266
267     elif choice == "SignUp":
268         st.subheader("Create New Account")
269         cust_name = st.text_input("Name")
270         cust_password = st.text_input("Password",
type='password', key=1000)

```

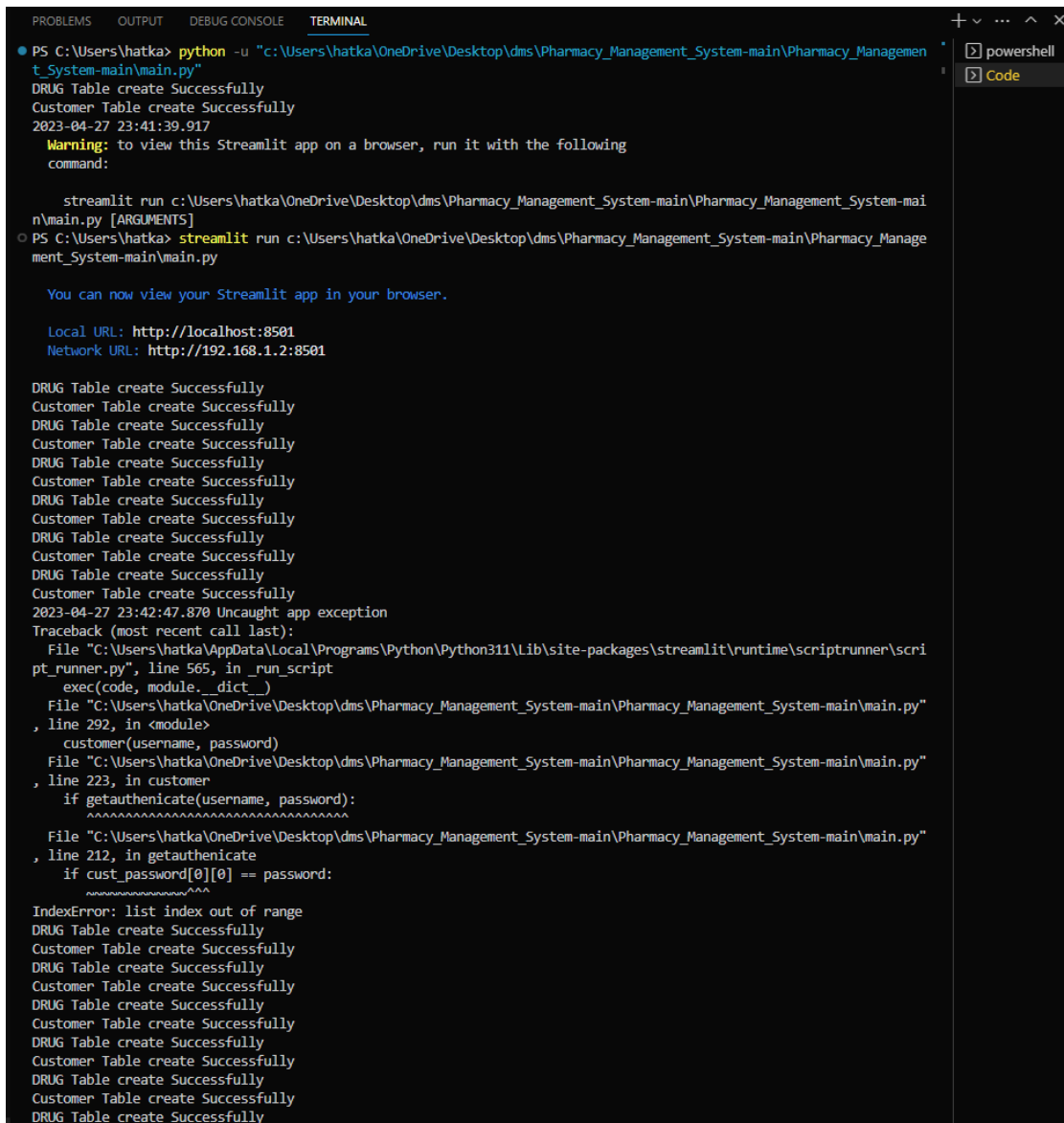


```

271     cust_password1 = st.text_input("Confirm
Password", type='password', key=1001)
272     col1, col2, col3 = st.columns(3)
273
274     with col1:
275         cust_email = st.text_area("Email ID")
276     with col2:
277         cust_area = st.text_area("State")
278     with col3:
279         cust_number = st.text_area("Phone
Number")
280
281     if st.button("Signup"):
282         if (cust_password == cust_password1):
283             customer_add_data(cust_name,cust_password,cust_email
, cust_area, cust_number,)
284             st.success("Account Created!")
285             st.info("Go to Login Menu to login")
286         else:
287             st.warning('Password dont match')
288     elif choice == "Admin":
289         username = st.sidebar.text_input("User Name")
290         password = st.sidebar.text_input("Password",
type='password')
291         # if st.sidebar.button("Login"):
292         if username == 'admin' and password ==
'admin':
293             admin()

```

5.2 TESTING



```
PS C:\Users\hatka> python -u "c:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py"
DRUG Table create Successfully
Customer Table create Successfully
2023-04-27 23:41:39.917
Warning: to view this Streamlit app on a browser, run it with the following
command:

streamlit run c:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py [ARGUMENTS]
PS C:\Users\hatka> streamlit run c:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.2:8501

DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
2023-04-27 23:42:47.870 Uncaught app exception
Traceback (most recent call last):
  File "C:\Users\hatka\AppData\Local\Programs\Python\Python311\Lib\site-packages\streamlit\runtime\scriptrunner\script_runner.py", line 565, in _run_script
    exec(code, module.__dict__)
  File "C:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py", line 292, in <module>
    customer(username, password)
  File "C:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py", line 223, in customer
    if getauthenticate(username, password):
    ~~~~~^~~~~~
  File "C:\Users\hatka\OneDrive\Desktop\dms\Pharmacy_Management_System-main\Pharmacy_Management_System-main\main.py", line 212, in getauthenticate
    if cust_password[0][0] == password:
    ~~~~~^~~~~~
IndexError: list index out of range
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
Customer Table create Successfully
DRUG Table create Successfully
```


6 RESULT & DISCUSSION

Hence we have successfully implemented medical store management system, A medical store management system is an essential tool for managing the day-to-day operations of a medical store or pharmacy. By automating inventory management, sales transactions, customer and supplier management, and reporting and analytics, a medical store management system can help improve efficiency, reduce errors, and save time for staff.

One of the key advantages of a medical store management system is the ability to track inventory levels and automate reordering. This can help prevent stockouts and reduce waste by ensuring that the store has the right amount of inventory on hand at all times. Additionally, features such as barcode scanning and automated billing can help reduce errors and improve the speed of transactions, leading to a better customer experience.

Another important feature of a medical store management system is the ability to generate reports and provide insights into the store's performance. By analyzing sales, inventory, and financial data, store owners and managers can make data-driven decisions and identify opportunities for growth and improvement.

One of the challenges of implementing a medical store management system is the initial setup and integration with existing systems. Depending on the size and complexity of the store, it may take some time to configure the system and train staff on how to use it effectively. Additionally, integrating the system with other systems such as EHRs and PBMs may require additional development work.

Overall, a medical store management system can be a valuable investment for any medical store or pharmacy looking to improve efficiency, reduce errors, and provide better customer service. By automating key processes and providing insights into performance, a medical store management system can help store owners and managers make better decisions and grow their business.

7 CONCLUSION AND FUTURE ENHANCEMENTS

The medical store management helps in managing resources and time. As technology continues to advance, there are several future enhancements that could be made to medical store management systems to improve their functionality and provide even greater value to medical stores and pharmacies. Here are some potential future enhancements:

1. **Integration with telemedicine platforms:** As telemedicine becomes more prevalent, integrating medical store management systems with telemedicine platforms could help streamline the prescription fulfillment process. This would allow physicians to send prescriptions directly to the medical store or pharmacy, reducing the need for manual data entry and improving the speed and accuracy of prescription fulfillment.
2. **AI-powered inventory management:** Using artificial intelligence (AI) and machine learning algorithms, medical store management systems could analyze sales data, seasonality trends, and other factors to predict inventory needs and automatically reorder items as needed. This would help reduce waste and ensure that the store always has the right amount of inventory on hand.
3. **Enhanced reporting and analytics:** While medical store management systems already provide valuable reporting and analytics features, future enhancements could include more detailed insights into customer behavior, product performance, and market trends. This would help store owners and managers make more informed decisions and identify new opportunities for growth and expansion.
4. **Mobile app integration:** Many customers now expect to be able to order products and manage their accounts through mobile apps. Integrating medical store management systems with mobile apps could help improve the customer experience by allowing customers to browse products, place orders, and view their prescription history on their mobile devices.
5. **Integration with blockchain technology:** Blockchain technology could be used to create a secure, transparent, and tamper-proof system for tracking drug supply chains. By integrating medical store management systems with blockchain technology, medical stores and pharmacies could help ensure that the drugs they receive are genuine and safe.

These are just a few potential future enhancements to medical store management systems. As technology continues to evolve, we can expect to see even more innovations that improve the efficiency and effectiveness of these systems.

8 REFERENCES

<https://www.scribd.com/document/365348816/Medical-Store-management-system>

<https://meeraacademy.com/use-case-diagram-for-medical-store-management-system/>

http://103.47.12.35/bitstream/handle/1/1566/1713104065_AashiJain_FinalProjectReport%20-%20Manoj%20Jain%281%29.pdf?sequence=1&isAllowed=y

<https://repo.ijert.org/index.php/ijert/article/view/3312>

https://www.researchgate.net/publication/274196269_Improving_medical_stores_management_through_automation_and_effective_communication

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4723708/>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4723708/>

https://www.academia.edu/31671486/Medical_store

<https://chat.openai.com/?model=text-davinci-002-render>

<https://sites.google.com/site/ignoubcafinalyearprojects/project-report/medical-store-management-system?pli=1>

<https://www.irjmets.com/uploadedfiles/paper/volume 3/issue 12 december 2021/18007/final/fin irjmets1641021488.pdf>

<https://www.ijircst.org/DOC/10-efficient-medical-management-system-for-pharamcist.pdf>

<https://gsconlinepress.com/journals/gscbps/sites/default/files/GSCBPS-2022-0343.pdf>