



Mukesh Patel School of Technology & Management Engineering

Vile Parle (w), Mumbai- 400056

Program: BTech CSDS(311)

Semester III

Report: IPP

Pharmacy Management System

By:

Ronak Bafna L002

Parth Dhadke L006

Soham Joshi L013

Jayaditya Modgil l016

Faculty In-charge: Pankti Doshi

Introduction

The efficient management of pharmacy operations is crucial in the healthcare sector, where timely and accurate service directly impacts patient care. The Pharmacy Management System project aims to simplify and streamline these operations by providing a software solution tailored specifically for pharmacies. This system is designed to manage essential functions, including product and customer records, transaction processing, and inventory tracking.

The project utilizes Python to create a user-friendly interface, employing essential libraries like Tkinter for GUI design and using data storage formats such as CSV to maintain persistent records. By incorporating data visualization tools, the system can generate insightful reports on total sales, top-selling products, and other critical metrics. Additionally, the software includes features like error handling, authentication, and data validation to enhance system security and reliability.

Our Pharmacy Management System simplifies day-to-day operations, reduces human errors, and facilitates data-driven decision-making in the pharmaceutical industry. Through this project, we explore how technology can improve operational efficiency in pharmacies, providing a robust foundation for inventory control, customer service, and sales analytics.

Problem statement

Pharmacies face numerous challenges in managing daily operations, including maintaining accurate inventory records, processing sales efficiently, and ensuring secure access to sensitive data. Traditional, manual approaches to pharmacy management often lead to errors in stock tracking, delays in serving customers, and limited insights into sales patterns. These inefficiencies can result in financial losses, stockouts of critical medicines, and reduced customer satisfaction.

Executive summary

The Pharmacy Management System is a Python-based application designed to streamline pharmacy operations, ensuring efficient management of medicine inventory, sales transactions, and essential data analysis. Built with Tkinter for the user interface and CSV files for data storage, this system supports secure user authentication with role-based access, allowing admins and staff to perform specific functions according to their permissions.

Key features include automated inventory management with low-stock alerts, medicine sales processing, and automated data backups to safeguard data integrity. Additionally, the system enables data-driven decision-making by analyzing top-selling medicines, calculating statistics on medicine pricing and stock levels, and visualizing sales trends with interactive charts. By automating repetitive tasks like inventory updates and sales logging, the system increases accuracy and efficiency in pharmacy operations, offering a reliable solution for enhancing productivity and supporting informed business decisions.

Data Description

The Pharmacy Management System uses two main CSV files to store and manage essential data: `Medicine_Database.csv` and `Sales_Records.csv`. Each file has a distinct purpose and structure designed to support the system's operations efficiently.

Medicine_Database.csv:

- **Purpose:** Stores key information for each medicine in the inventory, allowing the system to manage and track stock effectively.
- **Fields:**
 - *Medicine Name:* Unique identifier for each medicine.
 - *Price:* Unit price of the medicine, used in transaction calculations.
 - *Stock Quantity:* Current stock levels, which are adjusted as medicines are sold or replenished.
 - *Prescription Needed:* A Boolean field indicating whether a prescription is required. If True, doctor details are collected during the sale process for compliance.

Sales_Records.csv:

- **Purpose:** Keeps a record of each sale, allowing for transaction history tracking and sales analysis.
- **Fields:**
 - *Medicine Name:* Links each sale to a specific medicine.
 - *Quantity Sold:* Number of units sold, used to adjust stock levels and track demand.
 - *Total Price:* Total cost of the sale, calculated as $\text{Quantity Sold} * \text{Price}$.
 - *Sale Date:* Date and time of the sale, essential for analyzing sales trends.
 - *Customer Name and Customer Contact:* Information recorded for reference or follow-up if needed.
 - *Doctor Name and Doctor Address:* Required only if a prescription is necessary, ensuring regulatory compliance.

System Overview

The Pharmacy Management System is a comprehensive Python application designed to simplify and optimize pharmacy operations. The system integrates essential functionalities such as inventory management, sales processing, data analytics, and role-based access control to provide an effective solution for pharmacy staff and management. Key aspects of the system are as follows:

System Architecture

- **Frontend:** Developed using Tkinter, the graphical user interface (GUI) handles user input, displays information, and manages interactions through various windows and input dialogs, ensuring a user-friendly experience.

- **Backend:** Composed of data processing functions that manage reading, writing, and updating CSV files, this layer also implements key features for inventory analysis, backup, and analytics.
- **Data Storage:** Inventory and sales records are stored in two CSV files (`Medicine_Database.csv` and `Sales_Records.csv`). This choice allows for straightforward data manipulation and retrieval without complex database management.

User Roles and Permissions

The system incorporates a role-based access control model, distinguishing between two user roles: Admin and Staff.

- **Admin:** Has full access to all functionalities, including adding new medicines, updating stock, selling medicines, generating inventory reports, viewing sales records, and analyzing top-selling medicines.
- **Staff:** Has restricted access, limited to selling medicines, checking stock levels, viewing sales records, and generating inventory reports.

This access control enhances security by ensuring that only authorized users can perform certain sensitive actions, such as adding medicines or accessing analytics.

Core Functionalities

1. **Inventory Management:** Admins can add new medicines, update stock quantities, and check current stock levels. The system flags medicines that fall below a specified threshold to avoid stockouts.
2. **Sales Processing:** Both admins and staff can record sales transactions, capturing details such as medicine name, quantity sold, customer details, and doctor information if a prescription is required.
3. **Inventory and Sales Reports:** The system generates reports on current inventory levels, flagging low-stock medicines and summarizing recent sales activities.
4. **Data Analysis:** Analyzes top-selling medicines, calculates statistics on prices and stock levels, and visualizes sales data.
5. **Automated Backup:** Creates backups to ensure data preservation, making recovery possible in the event of data corruption or accidental deletion.

Data Flow and Processing

Each action (e.g., adding stock, recording a sale) triggers data updates in real time, ensuring accuracy across modules. This robust, user-friendly system addresses common challenges in pharmacy management by automating processes, enforcing role-based security, and providing insights through data analysis.

Data Handling and Processing

The Pharmacy Management System maintains data accuracy and reliability through structured handling and validation processes.

1. **Reading Data:** The system reads from `Medicine_Database.csv` and `Sales_Records.csv` using the Pandas library, ensuring quick and efficient data retrieval.
2. **Updating Data:** Inventory updates and sales are logged immediately in their respective CSV files, reflecting real-time changes.
3. **Data Validation:**
 - *Stock Validation:* The system checks for sufficient stock before completing a sale.
 - *Data Type Checks:* Price and stock quantities are checked for numeric values, preventing data entry errors.
 - *Prescription Requirement:* For medicines requiring a prescription, doctor details are mandatory, ensuring compliance.
 - *Low-Stock Alerts:* Items with low stock trigger alerts, notifying the admin for reorder.
4. **Backup System:** Automated backups are stored in a dedicated “backup” folder, providing a reliable safeguard for essential records.

System Functionalities

The Pharmacy Management System provides a range of functionalities to streamline pharmacy operations, enhance data accuracy, and facilitate role-based access control. Key system functionalities are outlined below:

1. User Authentication and Access Control

- The system employs a secure login feature to restrict access to authorized users. Each user is assigned a role—**Admin** or **Staff**—with specific permissions based on their responsibilities.
- **Admin** users have full access to system functionalities, including managing inventory, viewing sales records, analyzing top-selling medicines, and running reports.
- **Staff** users have restricted access and can perform essential tasks such as selling medicines, viewing stock levels, and generating inventory reports.
- This role-based access control enhances security and ensures that sensitive operations are performed only by authorized personnel.

2. Inventory Management

- **Add Medicine:** Admin users can add new medicines to the inventory by entering the medicine’s name, price, initial stock quantity, and prescription requirement status. This function updates the `Medicine_Database.csv` file with the new entry.
- **Update Stock:** Admins can update the stock of existing medicines by adding additional quantities. This feature ensures that the system reflects current inventory levels and prevents stockouts.
- **Low-Stock Alerts:** During inventory checks, the system flags medicines with stock quantities below a predefined threshold, alerting the admin to reorder critical items.

3. Stock Checking and Inventory Reporting

- **Check Stock:** Both Admin and Staff users can check the stock levels of any medicine in the inventory. The system displays the current stock quantity, highlighting medicines with low stock to prompt restocking.
- **Generate Inventory Report:** The system generates an inventory report that provides an overview of each medicine's name, price, stock quantity, and low-stock status. This report enables quick reference and is essential for managing inventory efficiently.

4. Sales Processing

- **Sell Medicine:** Both Admin and Staff users can record sales transactions, entering the medicine name, quantity sold, and customer details.
- **Prescription Verification:** For medicines requiring a prescription, the system prompts the user to enter doctor details (name and address) to ensure compliance with regulations. This information is stored in the sales record for reference.
- **Sales Logging:** Each sale is recorded in the `Sales_Records.csv` file, capturing details such as medicine name, quantity sold, total price, sale date, customer information, and (if applicable) doctor details. This feature ensures accurate and complete sales tracking.

5. Data Analysis

- **Analyze Top-Selling Medicines:** The system allows Admin users to view the top-selling medicines based on total quantities sold. This analysis is generated from sales data, enabling managers to identify high-demand items and make informed stocking decisions.
- **Calculate Price and Stock Statistics:** The system calculates key statistics, including the average, median, and standard deviation of prices and stock quantities. These metrics provide insights into inventory value and stock distribution, supporting data-driven inventory management.

6. Data Visualization

- **View Top-Selling Medicines Chart:** For a visual representation of demand, the system generates a bar chart displaying the top five selling medicines based on quantity sold. This chart aids in identifying popular items and informs restocking and sales strategies.
- The chart is generated using the Matplotlib library and displayed directly within the system interface, providing an accessible, data-driven visualization of sales performance.

7. Automated Data Backup

- The system includes an automated backup feature that periodically creates copies of the `Medicine_Database.csv` and `Sales_Records.csv` files after critical operations, such as adding stock or recording sales.
- **Backup Folder:** Backups are stored in a designated "backup" folder, ensuring data safety and providing easy access to previous records.

- This feature prevents data loss and supports recovery in case of accidental deletion or data corruption, offering a reliable safeguard for essential records.

Code Implementation and Analysis

The system is primarily built around two core classes, `PharmacyApp` and `Medicine`, which implement major functionalities including adding medicines, updating stock, selling medicines, and generating reports.

1. **PharmacyApp Class:** The main class that manages the GUI, handles user interactions, and controls the overall application flow.
2. **User Authentication:** Grants specific functionalities to users based on their role.
3. **Medicine Management:** Enables adding new medicines and updating stock quantities.
4. **Selling Medicine:** Processes medicine sales, updates inventory, and records transactions in `Sales_Records.csv`.
5. **Data Analysis and Visualization:** Provides insights into top-selling medicines and calculates price/stock statistics using Pandas and Matplotlib.
6. **Backup Functionality:** Ensures data safety through automated backups, stored in a designated folder.

Libraries Used

- **Tkinter:** Used for creating the graphical user interface (GUI).
- **Pandas:** Utilized for handling the medicine inventory and sales data stored in CSV files.
- **Matplotlib:** Employed for visualizing sales data, such as top-selling medicines.

Key Classes and Functions

The system is primarily built around two core classes: `PharmacyApp` (which manages the main application flow) and `Medicine` (which handles individual medicine data). These classes implement the major functionalities of the system, including adding medicines, updating stock, selling medicines, and generating reports.

PharmacyApp Class

The `PharmacyApp` class is the main class for the application. It initializes the GUI and controls user interactions.

```

34 class PharmacyApp(tk.Tk):
35     def __init__(self):
36         super().__init__()
37         self.title("Pharmacy Management System")
38         self.geometry("600x600")
39         self.configure(bg="lightblue")
40
41         self.role = None
42         self.permissions = set()
43
44         self.create_widgets()

```

Figure 1: The PharmacyApp class

In this code, the `PharmacyApp` class inherits from `tk.Tk` and initializes the application window. The `create_widgets` method sets up the login interface, while the role-based access control determines which actions the user can perform once logged in.

User Authentication

The authentication system checks the username and password against predefined credentials. If successful, it grants the user access to specific functionalities based on their role (either admin or staff).

```

61 def authenticate_user(self):
62     username = self.username_entry.get().strip().lower()
63     password = self.password_entry.get().strip()
64
65     if username in USER_CREDENTIALS and USER_CREDENTIALS[username] == password:
66         self.role = username
67         self.permissions = ACCESS_PERMISSIONS[username]
68         self.login_frame.pack_forget()
69         self.show_main_menu()
70     else:
71         messagebox.showerror("Login Error", "Incorrect username or password.")

```

Figure 2: User Authentication

Here, the `authenticate_user` function retrieves the username and password entered by the user, compares them to a dictionary of valid credentials (`USER_CREDENTIALS`), and, if valid, grants access to the main menu.

Medicine Management

The system allows users to add new medicines and update the stock levels. Here's how the `add_medicine` and `update_stock` methods are implemented:


```

102     def add_medicine(self):
103         name = simpledialog.askstring("Add Medicine", "Enter medicine name:").capitalize()
104         price = simpledialog.askfloat("Add Medicine", "Enter price:")
105         stock = simpledialog.askinteger("Add Medicine", "Enter stock quantity:")
106         prescription_needed = simpledialog.askstring("Add Medicine", "Prescription needed (yes/no:).strip().lower() == "yes"
107
108         new_medicine = Medicine(name, price, stock, prescription_needed)
109
110         df = pd.read_csv(MEDICINE_DB)
111         new_entry = pd.DataFrame([new_medicine.name, new_medicine.price, new_medicine.stock, prescription_needed]),
112         columns=['Medicine Name', 'Price', 'Stock Quantity', 'Prescription Needed'])
113         df = pd.concat([df, new_entry], ignore_index=True)
114         df.to_csv(MEDICINE_DB, index=False)
115         messagebox.showinfo("Success", f"Medicine '{new_medicine.name}' added successfully.")
116         auto_backup()

```

Figure 3: The add_medicine function

The add_medicine function captures the details of a new medicine, adds it to the Medicine_Database.csv, and displays a success message. Similarly, the update_stock function allows users to increase the stock of existing medicines.

```

118     def update_stock(self):
119         name = simpledialog.askstring("Update Stock", "Enter medicine name:").capitalize()
120         additional_stock = simpledialog.askinteger("Update Stock", "Enter additional stock quantity:")
121
122         df = pd.read_csv(MEDICINE_DB)
123         if name in df['Medicine Name'].values:
124             df.loc[df['Medicine Name'] == name, 'Stock Quantity'] += additional_stock
125             df.to_csv(MEDICINE_DB, index=False)
126             messagebox.showinfo("Success", f"Stock updated successfully for {name}.")
127         else:
128             messagebox.showerror("Error", f"Medicine '{name}' not found.")
129         auto_backup()

```

Figure 4: The Update_Stock Function

Selling Medicine

The process of selling medicine includes verifying if the item is in stock, updating inventory, and recording the sale in the Sales_Records.csv file.

```

150     def sell_medicine(self):
151         name = simpledialog.askstring("Sell Medicine", "Enter medicine name to sell:").capitalize()
152         quantity = simpledialog.askinteger("Sell Medicine", "Enter quantity to sell:")
153         customer_name = simpledialog.askstring("Sell Medicine", "Enter customer name:")
154         customer_number = simpledialog.askstring("Sell Medicine", "Enter customer contact number:")
155
156         df = pd.read_csv(MEDICINE_DB)
157         if name in df['Medicine Name'].values:
158             stock = df.loc[df['Medicine Name'] == name, 'Stock Quantity'].values[0]
159             price = df.loc[df['Medicine Name'] == name, 'Price'].values[0]
160             prescription_needed = df.loc[df['Medicine Name'] == name, 'Prescription Needed'].values[0]
161
162             if prescription_needed:
163                 doctor_name = simpledialog.askstring("Sell Medicine", "Enter doctor's name for prescription:")
164                 doctor_address = simpledialog.askstring("Sell Medicine", "Enter doctor's address:")
165             else:
166                 doctor_name, doctor_address = None, None
167
168             medicine = Medicine(name, price, stock, prescription_needed)
169             total_price = medicine.sell(quantity)
170
171             if total_price is not None:
172                 df.loc[df['Medicine Name'] == name, 'Stock Quantity'] -= quantity
173                 df.to_csv(MEDICINE_DB, index=False)
174
175                 sales_df = pd.read_csv(SALES_DB)
176                 new_sale = pd.DataFrame([name, quantity, total_price, datetime.now(), customer_name, customer_number, doctor_name, doctor_address]),
177                 columns=['Medicine Name', 'Quantity Sold', 'Total Price', 'Sale Date', 'Customer Name', 'Customer Contact',
178                 'Doctor Name', 'Doctor Address'])
179                 sales_df = pd.concat([sales_df, new_sale], ignore_index=True)
180                 sales_df.to_csv(SALES_DB, index=False)
181
182                 messagebox.showinfo("Success", f"Sold {quantity} of {name}. Total price: Rs. {total_price:.2f}")
183                 auto_backup()
184             else:
185                 messagebox.showerror("Error", "Insufficient stock.")
186         else:
187             messagebox.showerror("Error", f"Medicine '{name}' not found.")

```

Figure 5: The Sell_Medicine Function

In this function, when a user sells a medicine, the stock is checked and updated, and the sale is recorded. If there is insufficient stock, an error message is shown.

Data Analysis and Visualization

For analyzing the sales data, the system includes functions to display sales trends and generate graphs. The following function visualizes the top-selling medicines:

```
220     def plot_top_selling_medicines(self):
221         sales_df = pd.read_csv(SALES_DB)
222         sales_df['Quantity Sold'] = pd.to_numeric(sales_df['Quantity Sold'], errors='coerce')
223         sales_df = sales_df.dropna(subset=['Quantity Sold'])
224         top_selling = sales_df.groupby('Medicine Name')['Quantity Sold'].sum().nlargest(5)
225         plt.figure(figsize=(8, 5))
226         top_selling.plot(kind='bar', color='skyblue')
227         plt.xlabel('Medicine Name')
228         plt.ylabel('Total Quantity Sold')
229         plt.title('Top 5 Selling Medicines')
230         plt.xticks(rotation=45)
231         plt.tight_layout()
232         plt.show()
```

Figure 6: The Plot_Top_Selling_Medicines Function

This function generates a bar chart of the top-selling medicines using the sales data stored in `Sales_Records.csv`.

Backup Functionality

The `auto_backup` function automatically backs up the medicine and sales records to a designated folder:

```
235     def auto_backup():
236         os.makedirs("backup", exist_ok=True)
237         pd.read_csv(MEDICINE_DB).to_csv("backup/Medicine_Database_backup.csv", index=False)
238         pd.read_csv(SALES_DB).to_csv("backup/Sales_Records_backup.csv", index=False)
239         print("Auto-backup completed.")
```

Figure 7: The Auto_Backup Function

- **Main Application Flow:** The `PharmacyApp` class handles the user interface and all user interactions, from logging in to performing operations on medicines and sales.
- **Data Handling:** All data is managed using Pandas, allowing for easy manipulation of CSV files for medicines and sales records.
- **Visualization:** The system employs Matplotlib to provide meaningful insights into the sales data, such as top-selling medicines.

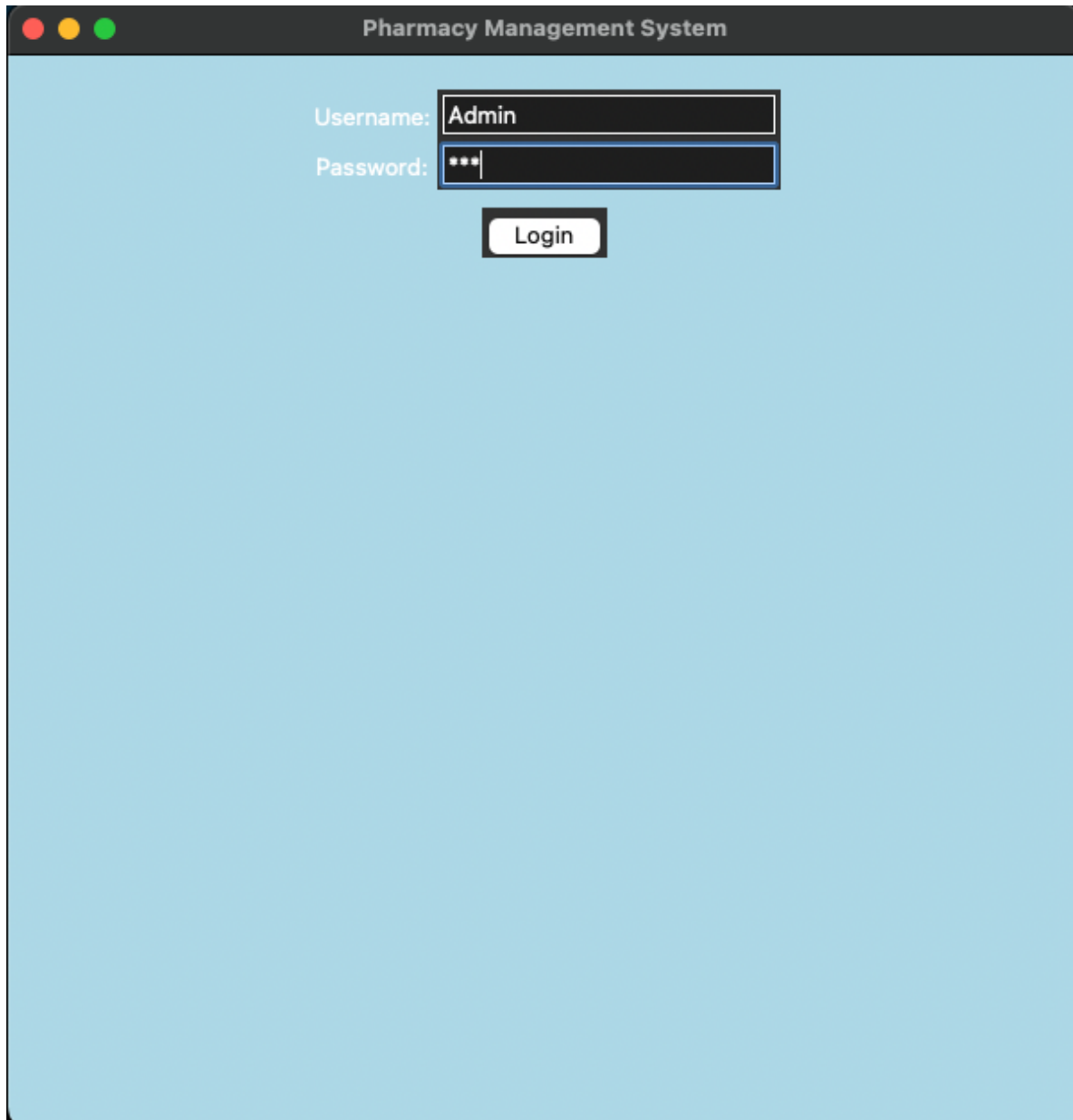
UI Structure

1. Login Screen

The first interface that appears when a user interacts with the system is the **Login Screen**. This screen ensures that only authorized users, such as admin and staff, can access the system. It contains fields for entering a **Username** and a **Password**.

- **Admin Authentication:** The system validates credentials entered by the user. If the login credentials are correct (for example, admin username and password), the user is granted access to the system's functionalities.
- **Error Handling:** If invalid credentials are entered, the system provides an error message, prompting the user to try again.

The login screen ensures the security of the system and restricts access to authorized personnel only.



The image shows a screenshot of a web application window titled "Pharmacy Management System". The window has a light blue background. In the center, there is a login form with two input fields: "Username:" and "Password:". The "Username:" field contains the text "Admin". The "Password:" field contains three dots, indicating a masked password. Below the input fields is a "Login" button. The window has a standard macOS-style title bar with red, yellow, and green window control buttons on the left.

Figure 8: Login screen

2. Main Menu

After a successful login, users are directed to the **Main Menu**. This menu serves as the control center of the system and provides access to all of the functionalities offered by the Pharmacy Management System. The menu is organized into clearly labeled options, each leading to a different section of the system.

The **Main Menu** displays the following key options:

- **Inventory Management:** To manage and view the list of medicines currently in stock.
- **Sales Records:** To track and record sales transactions.
- **Sales Report:** To generate reports based on sales data.
- **Backup Management:** To back up the system's data for security and recovery.
- **User Authentication:** For adding and managing users, including setting permissions for admin and staff roles.
- **Update Medicine:** To update the details of medicines in the inventory, such as price, quantity, and stock status.
- **Sell Medicine:** To record a sale of medicines, updating both the inventory and sales records.

Users can navigate through the menu by selecting the corresponding option to perform a specific task.

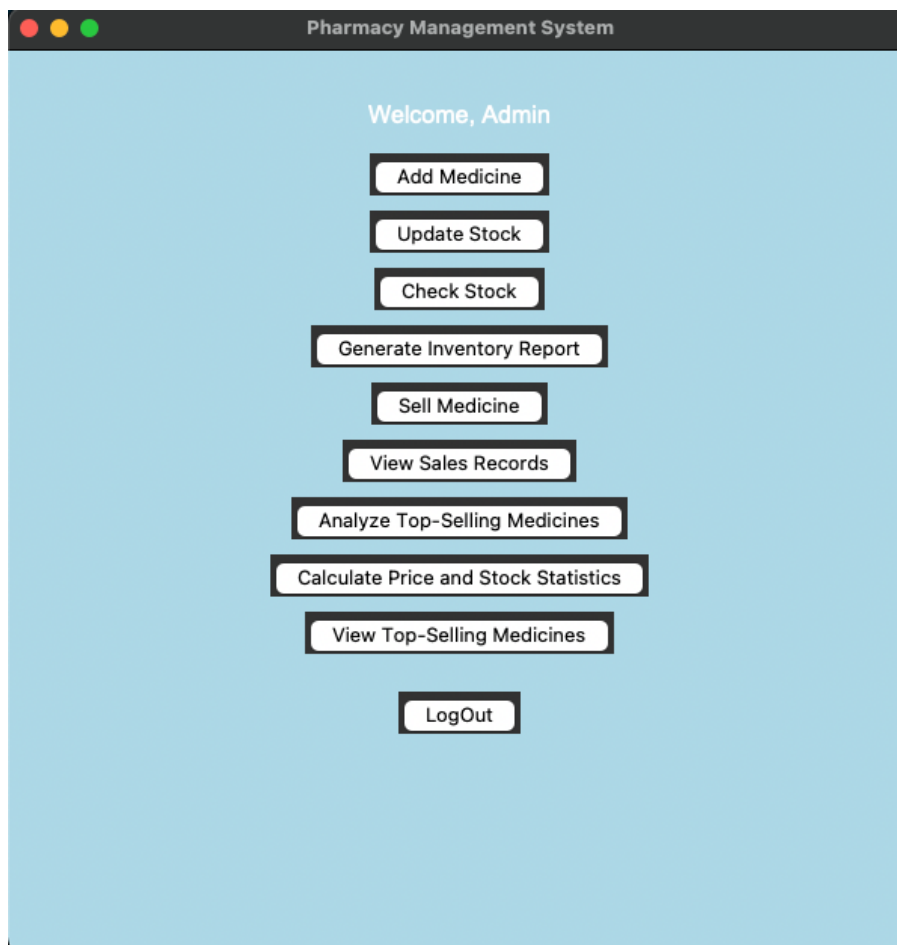


Figure 9: Admin Functions

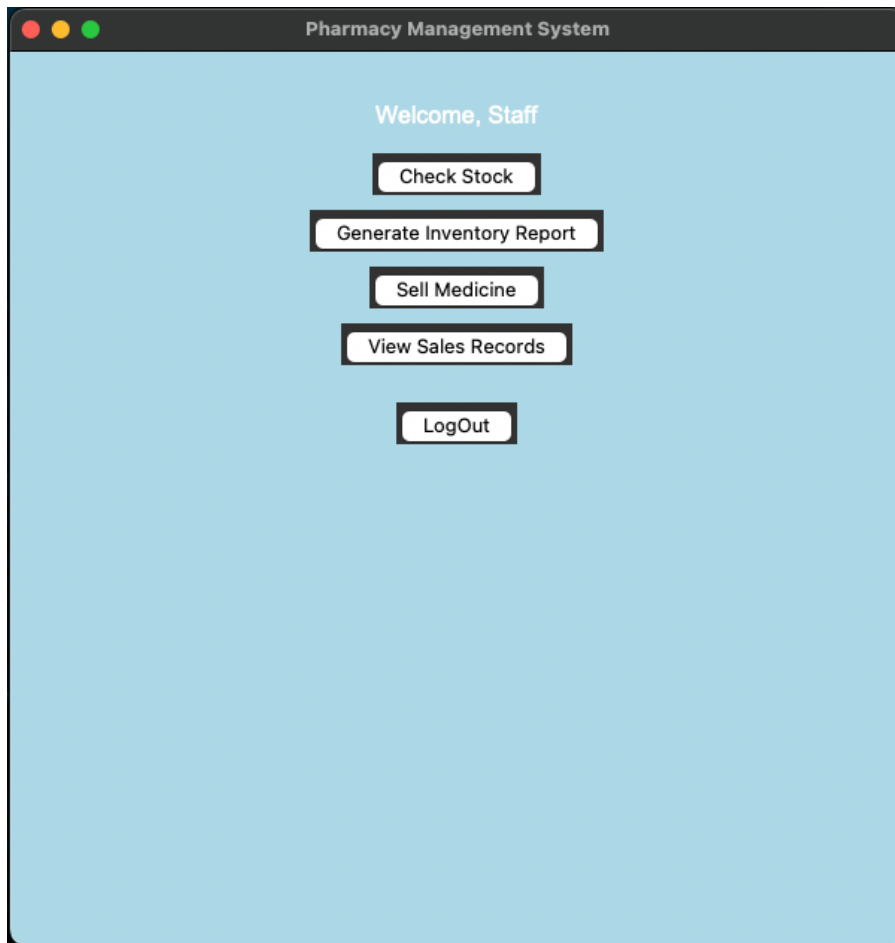


Figure 10: Staff Functions

Results and Analysis

This section highlights the key functionalities and results of the Pharmacy Management System.

1. **Check Stock Function:** Displays current stock levels and notifies users if quantities are below the threshold.
2. **Generate Inventory Report:** Provides an overview of all medicines with stock status, assisting in timely reordering.
3. **Sales Records Function:** Allows users to view past sales, tracking customer preferences and sales trends.
4. **Analyze Top-Selling Medicines:** Summarizes high-demand medicines, supporting inventory planning.
5. **Calculate Price and Stock Statistics:** Offers insights into inventory and pricing trends.
6. **View Top-Selling Medicine Function:** Visualizes popular items using a bar chart, aiding inventory and promotional planning.

Check Stock Function

The "Check Stock" function allows users to view the current inventory of a specific medicine. Upon entering the medicine name, the system checks the stock level from the database and displays the quantity. If the stock is below the defined threshold of 10 units, the system triggers a "Low Stock!" alert, notifying the user that restocking is needed. This feature is crucial for effective inventory management, ensuring that the pharmacy is aware of its stock levels and can take timely action to prevent shortages.

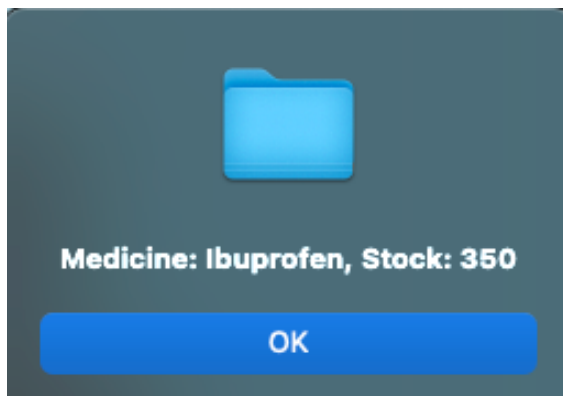


Figure 11: Checking stock for a specific medicine

Generate Inventory Report Function

The "Generate Inventory Report" function compiles a comprehensive report of the current stock of all medicines in the database. The system retrieves data, including the medicine name, price, stock quantity, and status (whether the stock is sufficient or low). For medicines with a stock quantity below the threshold of 10 units, the status is marked as "Low Stock!". The generated report is displayed to the user in a structured format, providing insights into the overall inventory and highlighting any items that may need restocking. This function is essential for maintaining accurate records and helping the pharmacy stay organized.




Medicine Name	Price	Stock
Medicine 1	353.38	41
In Stock		
Medicine 2	30.64	0
Low Stock!		
Medicine 3	330.59	47
In Stock		
Medicine 4	256.78	69
In Stock		
Medicine 5	331.65	48
In Stock		
Medicine 6	156.31	3
Low Stock!		
Medicine 7	64.20	49
In Stock		
Medicine 8	408.25	51

OK

Figure 12: Generating an inventory report showing stock status

Sales Records Function

The "Sales Records" function allows users to view detailed records of all completed sales transactions. The system retrieves information such as the sale date, medicine name, quantity sold, total price, and customer details (name and contact). This function provides a comprehensive overview of past sales, allowing the pharmacy to track sales trends, customer preferences, and overall revenue. It is vital for financial reporting, inventory planning, and customer service management.



Sale Date	Medicine Name	Quantity Sold	Total Price	Customer Name	Customer Contact	Doctor Name	Doctor Address
NaN	Medicine 167	7	1146.60	NaN	NaN	NaN	NaN
NaN	Medicine 249	7	2517.69	NaN	NaN	NaN	NaN
NaN	Medicine 65	5	2219.55	NaN	NaN	NaN	NaN
NaN	Medicine 56	11	1628.55	NaN	NaN	NaN	NaN

OK

Figure 13: Displaying detailed sales records

Analyze Top Selling Medicines Function

The "Analyze Top Selling Medicines" function provides a summary of the top-performing medicines based on sales quantity. The system aggregates sales data to identify the five medicines with the highest total quantity sold. This report helps the pharmacy understand which medicines are in high demand and provides insights for inventory planning and marketing strategies. By analyzing top-selling products, the pharmacy can focus on maintaining adequate stock levels and promoting these items to increase sales.

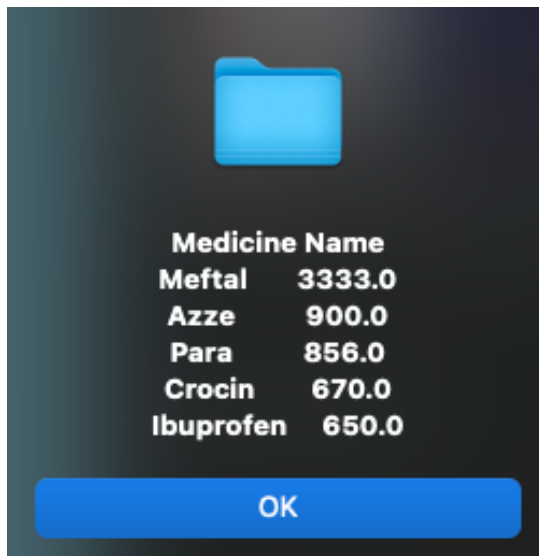


Figure 14: Analyzing top-selling medicines based on sales quantity.

Calculate Price and Stock Statistics Function

The "Calculate Price and Stock Statistics" function generates key statistics related to the prices and stock quantities of medicines in the inventory. The system calculates the mean, median, and standard deviation for both the prices and stock quantities. These statistics provide valuable insights into pricing trends, stock distribution, and inventory management. By understanding these metrics, the pharmacy can make informed decisions about pricing strategies, restocking needs, and overall inventory health.

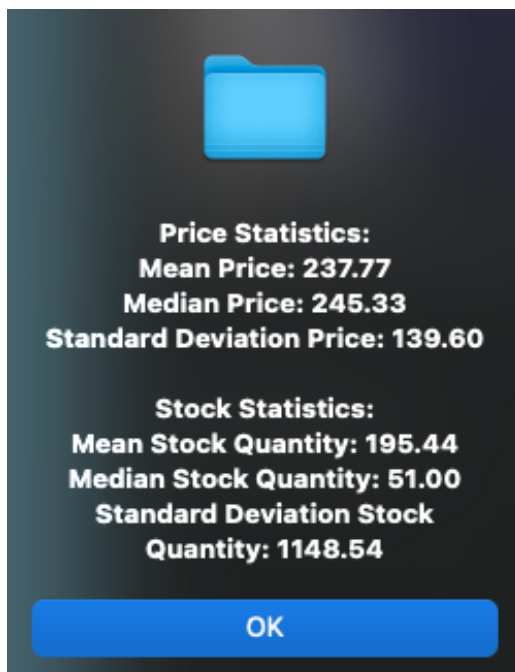


Figure 15: Calculating price and stock statistics.

View Top Selling Medicine Function

The "View Top Selling Medicine" function visualizes the top-selling medicines by plotting a bar chart of the total quantity sold for each medicine. The system aggregates sales data and highlights the top five medicines based on the highest sales volumes. This feature is useful for understanding sales trends and identifying popular products, which can guide inventory management and promotional efforts.

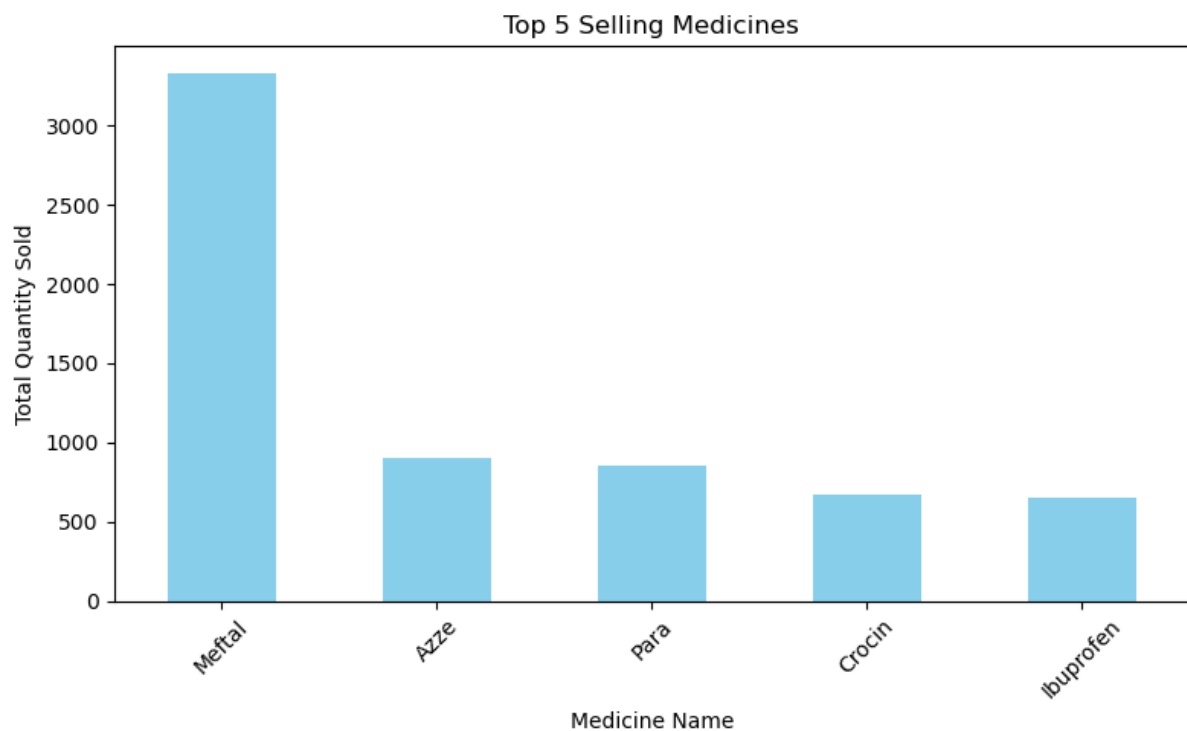


Figure 16: A Bar chart of the top-selling medicines based on total quantity sold.

Conclusion

The Pharmacy Management System successfully addresses key challenges in pharmacy operations by automating inventory management, streamlining sales processes, and providing insights through data analysis. The system's role-based access control enhances data security, and the automated backup function safeguards against data loss. The user-friendly interface, powered by Python libraries like Tkinter, Pandas, and Matplotlib, ensures accessibility and efficiency, enabling pharmacies to operate with greater accuracy and reliability. Overall, this system demonstrates the potential of technology to improve operational efficiency and support data-driven decision-making in the pharmaceutical industry.

Future Work

Future enhancements for the Pharmacy Management System could include:

1. **Predictive Analytics:** Incorporate machine learning algorithms to forecast high-demand medicines and optimize stock levels based on seasonal trends.

2. **Integration with Electronic Health Records (EHR):** Enable seamless communication with patient health records for improved customer service and prescription validation.
3. **Database Migration:** Transition to a SQL database for better scalability and more complex query handling as the system grows.
4. **Mobile Compatibility:** Develop a mobile version of the app to allow staff to manage inventory and process sales on-the-go.
5. **Reporting Dashboard:** Introduce an interactive dashboard that offers real-time updates on inventory, sales, and analytics, providing managers with a comprehensive view of operations.