Republic of the Philippines
Camarines Sur Polytechnic Colleges
Nabua, Camarines Sur
ISO 9001:2015

COLLEGE of COMPUTER STUDIES

Pharmacy Management System (PMS)

Group 5: Kaputt Kommandos

Prepared by:

Colico, Carl Renz M.
David, Kenji Nathaniel R.
Nonato, Francis Gabriel F.

Table of Contents

Project Overview & Problem Statement

The Kaputt Kommandos PMS is a modern desktop application designed to streamline daily pharmacy operations. It handles user authentication, inventory tracking, prescription validation, billing, and patient management. In this realm of the world, traditional pharmacy operations suffer from:

- Manual prescription handling leading to dispensing errors
- Paper-based inventory tracking causes stockouts and waste
- Fragmented patient records across multiple systems that juggle the operation
- Time-consuming billing processes that is time-consuming and prone to calculation errors
- Lack of an efficient management role is compromising data security

This project, Pharmacy Management System (PMS), is a modern desktop application that digitizes and streamlines pharmacy operations, centralizing medicine inventory with real-time stock monitoring, a digital prescription management system and validation workflow, a secure role-based access control for 6 user types, and a patient self-service portal for medicine browsing and orders.

The six (6) user types include:

Administration - System configuration and user management

Pharmacists - Prescription validation and medicine dispensing

Inventory Managers - Stock monitoring and reorder management

Billing Clerks - Invoice generation and payment processing

Staff Members - Data entry and patient assistance

Patients - Account creation, medicine browsing, order tracking

Feature List & Scope Table (what's in/out)

Implemented Features (v1.0)

Features	Descriptions	User Roles	Status
User Authentication	Login with username/password, patient self-registration	All	Complete
Role-Based Access Control	Dynamic dashboards and menus per role	All	Complete
Patient Dashboard	Overview of prescriptions, orders, and quick actions	Patient	Complete
Medicine Search & Browse	Search by name, view details, and stock availability	Patient	Complete
Shopping Cart	Add medicines to cart (UI ready)	Patient	Complete
Patient Profile	View/edit personal information, logout	Patient	Complete
Admin Dashboard	System statistics, quick actions, recent activity	Admin	Complete
User Management	Edit and delete users; role actions	Admin	Not Implemented
System Administration	5 report types (User Activity, Inventory, Prescriptions, Low Stock, Usage)	Admin	Complete
System Reports	View and Generate User Activity Reports	Admin	Complete
System Logs (IAS Enhancement)	View system actions, filter by type/date	Admin	Complete

Medicine Database	Store medicine details (name, category, price, stock, expiry)	Admin	Complete
Pharmacist Dashboard	Overview of the Pharmacist's Quick Actions (Search meds, Generate Report, Review prescriptions)	Pharmacist	Complete
View Prescription	View the prescription details for the patient (solo)	Pharmacist	Complete
Prescription Management	Upload, review, approve	Pharmacist Patient (can just upload)	Complete
Medicine Search	Search for Medicine based on the customer's prescription (category/stock)	Pharmacist	Complete
Dark/Light Mode	Theme toggle with Teal color scheme	All	Complete
Stock Monitoring	Low stock detection in the dashboard, notifications, and alerts	Pharmacist Admin Patient Inventory Manager	Complete
Inventory CRUD	Database + search functional. For the Inventory Manager, it should also add/edit/delete UI for the Inventory role	Inventory Manager Pharmacist	Complete
Medicine Dispensing	The pharmacist can dispense medical attribution, and the	Pharmacist Inventory Manager	Complete

	inventory manager can also provide the needed medicine for both patient and pharmacist (stock generation or change of medicine)		
Billing System	The bill of the patient can be accessed by the billing clerk, and the patient can review the bill and pay it.	Billing Clerk Patient	Completed
Create Prescription & Review	Patient & Pharmacist can Upload, review, approve.	Pharmacist Patient	Complete
Generate Report	Generate an overall statistics report of each role	Pharmacist Billing Clerk Admin Staff	Complete
Reporting & Analytics	Generate an analytics report performance of each role	Pharmacist Billing Clerk Admin Staff	
Session Timeout		All	Not Yet Implemented
Notifications	Have a system notification for all roles	All	Partially Implemented
Patient Log	Can view and see the system log of the patient activity	Staff Member Pharmacist	Completed
Real-time notifications system	An alert notification system on every dashboard for low stock medicine	All	Completed
Audit Log Viewer for bill and patient	Can audit the log activity as well as the payment	Billing Clerk Admin	Complete

	activity between the user and the billing clerk		
Audit log viewer for log report	Can generate a log system report for the overall system	Admin	Completed

FOR IAS FEATURE CHECKING (di to kasama sa documentation)

Minimum Functional Requirements (Baseline – must be implemented)

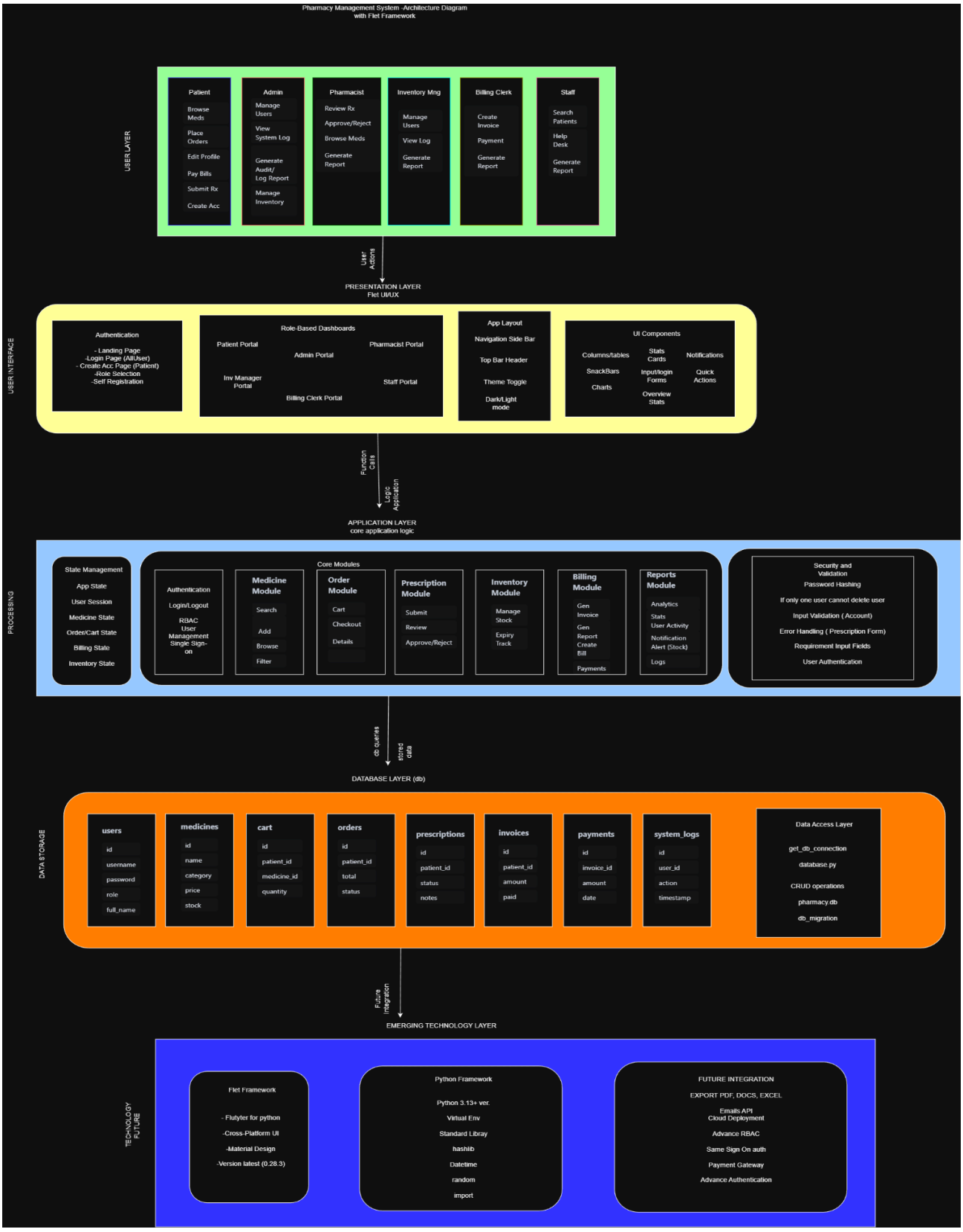
- ☑ ~~User Authentication~~
  ~~Role-Based Access Control (RBAC)~~
- ☑ ~~User Management (Admin only)~~
- ☑ ~~Profile Management (Self-service~~
- ☐ Security & Session Controls Data Layer
- ☑ ~~Logging (Baseline)~~
- ☐ Secure Configuration

ENHANCEMENT REQUIREMENTS AREAS

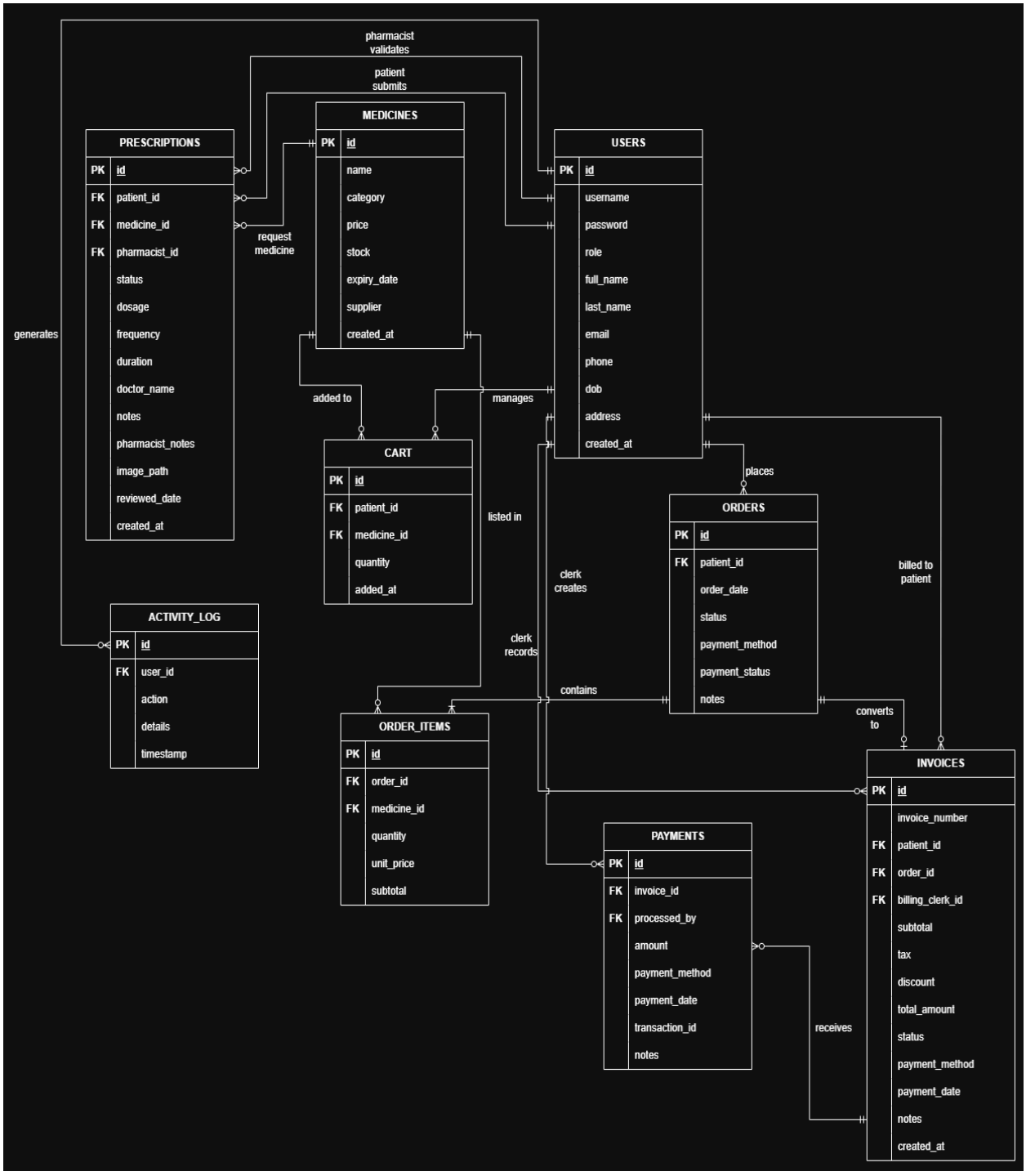
Pick and implement at **least 3**; more may earn a bonus if well-executed:

- ☐ Two-Factor Authentication (email, TOTP, or SMS gateway simulation)
- ☐ Password Policy (complexity, reuse prevention, expiration, breach check via k-Anonymity API, optional)
- ☐ Advanced RBAC (custom roles + permission matrix UI)
- ☑ ~~Audit Log Viewer (filter by actor, date range, action type)~~
- ☐ REST API (token-based or OAuth2) with least-privilege scopes
- ☐ Single Sign-On (OAuth2 / OIDC with Google or GitHub) – still retain local admin bootstrap
- ☑ ~~User Activity Monitoring (last login, failed attempts)~~
- ☐ Automated Backup & Restore (scheduled export + documented restore path)
- ☐ Multi-Tenancy (org separation: data scoping + admin boundaries)
- ☑ ~~Reporting & Analytics (charts for usage, role distribution)~~
- ☐ Secure Password Reset (signed time-bound token email flow)
- ☐ Secrets Vault Integration (e.g., environment abstraction, mocking Vault)

Architecture Diagram



Data Model (ERD)



Emerging Technology Integration

Primary Technology: Flet Framework

What is Flet?

- Python framework based on Flutter for building beautiful, cross-platform applications
- Allows Python developers to create native desktop, web, and mobile apps with a single codebase
- Uses Google's Material Design components for modern, consistent UI

Why Flet Was Chosen:

1. **Rapid Development**
    - Python-based: Leverages team's existing Python knowledge
    - Hot reload: Instant UI updates during development
    - Rich widget library: Pre-built Material Design components
2. **Cross-Platform Support**
    - Desktop: Windows, macOS, Linux
    - Web: Can be deployed as a web app
3. **Modern UI/UX**
    - Material Design 3 compliance
    - Built-in dark mode support
    - Responsive layouts
    - Smooth animations and transitions
4. **Low Barrier to Entry**
    - No need to learn Dart/Flutter
    - Familiar Python syntax
    - Active community and documentation

How Flet is Integrated:

```
# App Structure
import flet as ft
def main(page: ft.Page):
# Page configuration
   page.title = "Pharmacy Management System"
   page.theme = ft.Theme(color_scheme_seed=ft.Colors.TEAL)
```

```
    page.theme_mode = ft.ThemeMode.LIGHT # Dynamic routing
    page.on_route_change = route_handler page.go("/")

ft.app(target=main)
```

Key Features Utilized:

1. **Navigation System**
   - `ft.NavigationRail` for sidebar menu
   - Dynamic route-based page switching
   - Role-based navigation visibility

2. **Material Components**
   - `ft.Container`, `ft.Row`, `ft.Column` for layouts
   - `ft.ElevatedButton`, `ft.TextField` for inputs
   - `ft.DataTable`, `ft.Dropdown` for data display
   - `ft.AlertDialog`, `ft.SnackBar` for feedback

3. **Theming**
   - Centralized theme configuration
   - Dynamic theme switching (Light/Dark)
   - Consistent color palette (Teal-based)

4. **State Management**
   - Custom AppState class for user session
   - Event-driven updates with `page.update()`

Limitations:

1. **Desktop-First Focus**
   - Current version optimized for desktop (1024x768+)
   - Mobile responsiveness requires additional work

2. **Learning Curve for Advanced Features**
   - Complex animations require Flutter knowledge
   - Custom widgets need a deeper framework understanding

3. **Performance with Large Datasets**
   - UI may slow down with 1000+ list items
   - Pagination/virtualization needed for large tables

4. **Deployment Complexity**
    - Requires Python runtime on client machines
    - Web deployment needs server setup

Future Enhancements:

- Implement `ft.ListView` with lazy loading for medicine catalog
- Add `ft.ProgressRing` for async operations
- Utilize `ft.Tabs` for multi-section views
- Explore web deployment with the Flet web server

Setup & Run Instructions

Prerequisites

- **Python:** Version 3.10 or higher
- **pip:** Python package installer
- **Git:** For cloning the repository

Platform-Specific Notes

Windows:

- Python 3.13.5 recommended
- PowerShell or Command Prompt

macOS:

- Python 3.10+ (via Homebrew: brew install python@3.13)
- Terminal

Linux:

- Python 3.10+ (Ubuntu: sudo apt install python3.13)
- Terminal

1. Clone the Repository

```
git clone https://github.com/frnonato-lgtm/Pharmacy-Management-System.git
cd Pharmacy-Management-System
```

2. Create Virtual Environment: Windows (PowerShell):

```
python -m venv .venv
.\.venv\Scripts\Activate
```

Windows (Command Prompt):

```
python -m venv .venv
.venv\Scripts\activate.bat
```

macOS/Linux:

```
python3 -m venv .venv
source .venv/bin/activate
```

3. Install Dependencies

```
pip install -r requirements.txt
```

Dependencies installed:

- `flet==0.28.3` - UI framework
- `flet-core==0.28.3` - Core Flet library
- (SQLite3 is built into Python)

4. Initialize Database (Important!)

- You must run the migration script to create the tables and seed the database with the 54 medicines and default users.
    1. **Check for old data:** If a **pharmacy.db** file already exists in **src/storage/**, **delete it** to ensure a fresh start.
    2. **Run the seed script:**

```
python src/services/db_migration.py
```

You should see the message: "🎉 MIGRATION & SEEDING COMPLETED SUCCESSFULLY!"

5. Run the Application

```
cd src
python main.py
```

Default Login Credentials (New patients can self-register from the landing page)

Role	Username	Password

Administrator	admin	admin123
Pharmacist	pharm	pharm123
Inventory Manager	inv	inv123
Billing Clerk	bill	bill123
Staff Member	staff	staff123

Testing Summary

Authentication & Access Control:

- ☐ User logs in with valid credentials (all roles)
- ☐ Login rejection with invalid credentials
- ☐ Patient self-registration with form validation
- ☐ Role-based dashboard redirection
- ☐ Role-based menu visibility (Patient, Admin, etc.)
- ☐ Logout functionality from the profile page

Patient Portal:

- ☐ Medicine search by name
- ☐ Display medicine details (price, stock, category)
- ☐ Shopping cart interface (UI)
- ☐ Order history view (mock data)
- ☐ Profile information display
- ☐ Profile edit form (UI ready)

Admin Functions:

- ☐ User management (view, search, filter, delete)
- ☐ System statistics display (users, medicines, stock alerts)
- ☐ Report generation (5 types)
- ☐ Activity logs with filtering
- ☐ Low stock alerts on the dashboard

UI/UX:

- ☐ Dark mode / Light mode toggle
- ☐ Responsive layout at 1024x768

- ☐ Navigation consistency across roles
- ☐ Material Design compliance
- ☐ Error message display

Inventory Management:

- ☐ Medicine database queries
- ☐ Stock level display
- ☐ Add/Edit/Delete medicine operations (UI not implemented)
- ☐ Stock deduction on order completion

Prescription Workflow:

- ☐ Database schema functional
- ☐ Upload prescription image
- ☐ Pharmacist review interface
- ☐ Approval/rejection workflow

Billing System:

- ☐ Invoice database schema
- ☐ Invoice generation from cart
- ☐ Payment processing
- ☐ Receipt printing

How to Run Tests:

1. **Setup**

```
cd src
python main.py
```

2. **Test Authentication:**
    - Try logging in with admin/admin123
    - Try logging in with the wrong password (should fail)
    - Register a new patient account
    - Verify redirection to the correct dashboard
3. **Test Patient Flow:**
    - Log in as pat/pat123
    - Navigate to "Search Meds"

- Search for "Paracetamol"
- View medicine details
- Navigate to "My Profile"
- Click "Logout"

4. **Test Admin Functions:**
   - Log in as admin/admin123
   - View dashboard statistics
   - Navigate to "Users" → search for users
   - Navigate to "Reports" → generate Inventory Status Report
   - Navigate to "Logs" → view activity logs
5. **Test UI:**
   - Toggle dark/light mode (moon icon in top-right)
   - Verify consistent styling across pages
   - Test sidebar navigation

Known Issues:

- ☐ The Patient Prescription upload feature is non-functioning in the prescription_view.py (Removed)
- ☐ Patient billing feature is still incomplete (not yet connected to patient invoices_view)
- ☐ My Bills for patient billing are inoperative with invoices_view after the billing clerk creates the invoice
- ☐ User management (Admin): Add/Edit/Delete not implemented
- ☐ Profile Management (Patient): Edit Profile and Change Password details are still not working (Fixed)
- ☐ Fixed data issue with the patient added prescription to the prescription_view of the pharmacist
- ☐ Medicine database (Pharmacist): Update stock and view medicine details(can just remove these two buttons there)
- ☐ Export functionality not implemented (can just delete all export functionality throughout users' dashboards) (Removed)
- ☐ Staff Member, part the all patients' views, the search button is still buggy
- ☐ Submit Prescription Button not working in uploading prescription should generate a prescription form after clicking the button

- ☐ Remove the navigation back button for the pharmacist, billing clerk, and staff member
- ☐ Remove the export PDF/CSV or Export logs button for the admin in Logs and Reports.
- ☐ Remove the print button from All Invoices
- ☐ Help Desk of Staff Member adjust text content and icons
- ☐ Create Terms and Conditions

Test Environment:

- OS: Windows 11
- Python: 3.13.5
- Screen Resolution: 1366x768 and 1920x1080 (tested at 1024x768 window)

Future Testing Plans

Version 1.1:

- Automated unit tests using pytest
- Database integration tests
- Role permission boundary tests

Version 1.2:

- Load testing with multiple concurrent users
- Security penetration testing
- Accessibility testing (keyboard navigation, screen readers)

Version 2.0:

- Cross-platform testing (macOS, Linux)
- Mobile responsiveness testing
- Performance benchmarking

Team Roles & Contribution Matrix

Name	Primary Role	GitHub Username
Carl Renz M. Colico	Product Owner / PM	@carlcolico
Kenji Nathaniel R. David	QA Lead / Documentation	@kenjinathaniel

Francis Gabriel F. Nonato	Lead Developer	@frnonato-lgtm

Contribution Matrix

Components	Colico	David	Nonato	Notes
Requirements Gathering	45%	5%	50%	Case Matrix, SRS, Documentation, Feature Prioritization
UI/UX Design	45%	10%	45%	Wireframe, Architecture, Design System
Project Structure Setup	40%	0%	60%	Local and GitHub Repository Structure
Role-Based Dashboards, Overall Core Logic, and Function Components	60%	0%	40%	Presentation Layer, Logic UI/UX, Application Logic, and Core Modules
Presentation	33.33%	33.33%	33.33%	Presentation of the Project to the Instructor

Individual Responsibilities

Name: Carl Renz M. Colico
Owned Components: Patient, Billing Clerk, Admin, Staff, Pharmacist, Quick Actions
Key Contributions: Patient, Billing Clerk, Admin, Staff, Pharmacist Dashboard, and other UI/UX components, and Documentation

Github Stats: (no. of commits)

Name: Kenji Nathaniel R. David
Owned Components: Staff and Landing Page
Key Contributions: Wireframe, Documentation

Github Stats: (no. of commits)

Name: Francis Gabriel F. Nonato
Owned Components: Landing Page, Login Page, Project Structure
Key Contributions: Landing Page, Login Page, Project Structure, UI/UX components, and Documentation

Github Stats: (no. of commits)

Risk, Constraints & Future Enhancements

Risk

- The system must comply with the safeguarding of sensitive data information using encryption.
- Accessibility of medicine CRUD functionality must be restricted to pharmacists, inventory managers, and administrators.
- The system is likely to have limited space for handling information records.
- device (device used in the making) functions, but later on will depend on other or future measures.
- System development of the Project is developed using the Flet framework and Python, which limits its environment.
- Admin has higher accessibility in managing the system's features and other functions.
- The system interface can likely not be very responsive on other display resolutions or other systems.
- Some content has a limited responsive interface of 1024 x 768
- System development and testing are constrained by limited time.
- Future scalability or integration with hospital management systems is outside the current project scope, but may be considered in future versions.
- The project will or can be focused on local-based or network-based in the later or future phases.

- All users entering password data must have it encrypted
- The system must authenticate users through an app that states a single sign-on (SSO) service.
- Development must be completed using the Flet framework.
- Does not really apply the reality guidelines of pharmacy.

Constraints

Technical Constraints:

- The system is developed using the Flet framework (Python), limiting deployment to platforms that support the Python runtime.
- A SQLite database is used for data storage, which may have limitations for concurrent multi-user access.
- No external API integrations (payment gateways, insurance systems) in the current version. - - Limited to local/network deployment; cloud deployment not included in v1.3.

Security Constraints:

- User passwords are hashed using SHA-256 (Note: Consider upgrading to bcrypt in future versions).
- Role-based access control must be strictly enforced at all access points.
- Session management with automatic logout after inactivity.

Data Constraints:

- Database backups must be performed regularly (recommend daily automated backups).
- Patient data must comply with data privacy regulations.

Development Constraints:

- The development timeline is constrained by the academic semester schedule
- Testing is limited to desktop environments (Windows primarily).
- The user interface may not be fully responsive on all screen resolutions.

Future Enhancements

- Prescription Upload: Let patients upload photos of their prescriptions
- Medicine Refill Reminders: Remind patients to refill their medications
- Order History Export: Download order history as PDF
- Favorite Medicines: Quick reorder of frequently purchased items
- Medicine Interactions Checker: Warn if prescribed medicines conflict
- Shift Scheduling: Track who's working when
- Stock Transfer: Move stock between branches (if multiple locations)
- Two-Factor Authentication (2FA): Extra login security
- Automatic Backups: Daily database backups
- Session Timeout: Auto-logout after inactivity

- QR Code Generation: QR codes for prescriptions/orders
- Export to Excel/CSV: Export any data for analysis
- Multi-language Support: English, Filipino, etc.
- Keyboard Shortcuts: Speed up common actions
- Mobile-Responsive Design: Better mobile experience
- Doctor Notes: Store special instructions from doctors
- Receipt printing for better customer service

Individual Reflection (per member: 150–200 words)

Member Name: Carl Renz M. Colico
My Reflection:
Working on this project system is honestly overwhelming with the time we start it late I knew that this will be challenging as we decided to create a managing system especially we decided to manage a pharmacy, but I didn't expect that we would be creating bunch of interfaces, since this project purposely used Flet, and it is a new for us to use this on our project. The system components of users we decided are patient (user), billing clerk, admin, staff, inventory manager and a pharmacist. In the creating of the dashboard I realized that it really does have a lot of stuff and later on I was trapped or the hardest part was planning how the overall structure of each dashboard will connect properly to other dashboards or roles from clicking something all the way down to the database and back. Our team decided to approach layered UI on top, then business logic, then database stuff at the bottom. With flet, framework and modules availability this helps us a lot with making things look professional without spending forever on styling parts. After this was fixing bugs especially the CRUD part because it's always the button of add, update, create and delete part that always does not work on my part but thanks to my team we debugged it before the presentation. Overall, this project we have this time is a project poured with sleepless nights and dry lungs will forever I will remember and the lessons of how hard really is on deploying such systems or apps.

No.of words : 200+

Member Name: Kenji David Nathaniel
My Reflection:
This project as we gather information about our project we didn't plan to manage a system however we didn't knew that we would come into this system managing a pharmacy system, at first we didn't doubt that this would be hard but as my team on my group start it I did see that they are implementing not just this project have many roles it also have many components and logic that is really hard to tell. I work many times partly on documenting and other requirements, and as I can see that as the week goes by, I really can now somehow tell that this system is huge when it comes to its feature core components, and as my team builds it. It really makes me think about how they will

connect these components on each part, and I also heard from them that they really have a hard time adjusting some parts.

No.of words: 152

Member Name: Francis Gabriel F. Nonato
My Reflection:
Working on the Pharmacy Management System was a challenging but rewarding experience. At first, the project seemed overwhelming because of all the different user roles, like the Pharmacist, Admin, and Patient. However, breaking it down into specific views and services helped me understand how a real application is structured. I learned that coding the interface with Flet is only half the battle. The backend logic, like making sure the inventory updates automatically when a patient orders, was tricky but satisfying to solve.

The deployment process taught me the most. We faced issues with the database paths and git conflicts, especially when I accidentally tried to upload the huge executable file. Learning how to use gitignore and GitHub Releases properly saved the project. It made me realize that software development is not just about writing code but also about managing files and making sure the end user can actually run the program. Seeing the app fully functional on Windows with a working database gave me a lot of confidence in my programming skills. It was a great way to apply what we learned in class to a practical scenario.

No.of words: 188