UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

SEMESTER 1 2025/2026

BITU 3923 WORKSHOP 2

FINAL REPORT

SUPERVISOR: TS. NOR MAS AINA BINTI MD BOHARI

PROJECT TITLE : MEDICINE PRESCRIPTION MANAGEMENT SYSTEM (MPMS)

NAME	MATRIC NO
FARAH DAMIA BINTI MOHAMAD NIZAN	B032420034
ADAM BIN AZMI	B032410186
SUFIANA ADLIN BINTI BAHAROM	B032410816
IRFAH NADIAH BINTI HAMDAN	B032310418
MUHAMMAD SYAMEEL AMNI BIN MOHD SAIFUL AMRI	B032310661

TABLE OF CONTENTS

TABLE OF FIGURE

EXECUTIVE SUMMARY

The Medicine Prescription Management System (MPMS) is a database-oriented application designed to streamline and automate the critical operations of a pharmacy. The central objective is to transition from manual record-keeping and siloed systems, which currently lead to data duplication, inaccuracies, and inefficiency, to a cohesive, modernized platform. This system will emphasize the secure and effective management of medications, prescriptions, patient details, and sales data. A key technical specification is the MPMS's ability to demonstrate database interoperability and portability by being compatible with and utilizing PostgreSQL, MySQL, and Microsoft SQL Server. This cross-platform capability not only enhances operational flexibility but also serves as a valuable case study in effective database design and execution across various platforms.

The project's scope is defined by the development of five highly integrated modules designed to solve specific operational pain points. The Medicine Inventory Management Module will accurately track stock levels, expiry dates, and supplier information to eliminate stock shortages and errors. The Prescription Management Module is crucial for allowing pharmacists to efficiently create, edit, and manage prescriptions, thereby reducing the miscommunication and delays currently plaguing doctor-pharmacist interactions. Furthermore, the Patient Information Management Module will securely store and update comprehensive patient profiles and medical records, directly resolving issues of incomplete histories faced with manual storage. The system will also feature a Sales and Billing Module to automate transaction recording and invoice generation, making the checkout process faster and less error-prone.

Finally, the MPMS will integrate a powerful Reports and Analytics Module that generates comprehensive data on stock, sales trends, prescriptions, and overall system usage. This feature is vital for better decision-making and for the pharmacy to monitor performance, which is currently a challenge due to the lack of proper tools. The system is intended for use by Pharmacists, Patients, and System Administrators, making it a comprehensive solution for digital healthcare management. By implementing this system using HTML, CSS, JavaScript, and PHP, the team aims to deliver a modern application that not only improves the operational effectiveness of pharmacies but also enhances healthcare information management strategies by setting a standard for accuracy and platform flexibility.

CHAPTER 1: INTRODUCTION

1.1 Introduction

The Medicine Prescription Management System (MPMS) is a database-oriented application created to streamline and automate the functions of a pharmacy. It emphasizes the secure and effective management of medications, prescriptions, patient details, and sales data. In conventional pharmacies, many records are still managed manually or through unconnected systems, which often result in data inconsistencies, delays, and mistakes. This initiative seeks to develop a cohesive system that can function with various database systems, specifically PostgreSQL, MySQL, and SQL Server. This requirement is intended to showcase database interoperability and portability. The system will assist pharmacists, doctors, and patients by delivering precise, immediate access to information for managing prescriptions, monitoring inventory, and processing billing. The MPMS project aims to provide a modern solution that enhances healthcare information management and operational effectiveness.

1.2 Problem Statements

The Medicine Prescription Management System (MPMS) project is motivated by several critical issues present in conventional pharmacy operations that rely on manual record-keeping or unconnected systems. The primary problems this project seeks to resolve are:

- Inventory Errors and Stock Management Issues.

  Manual processes in recording and tracking medicine stock, expiry dates, and supplier details often cause inventory errors and stock shortages.

- Disintegrated Prescription Handling.

  The absence of an integrated prescription management system leads to confusion, delayed updates, and miscommunication between doctors and pharmacists.

- Incomplete Patient Records.

  Storing patient information manually results in incomplete records, making it difficult to retrieve accurate medical histories.

- Slow and Error-Prone Sales Process.

  The sales and billing process is slow and prone to human error due to the lack of an automated transaction recording system.

- Lack of Performance Monitoring.

  Without proper reporting and analytics tools, the pharmacy struggles to monitor sales trends, medicine performance, and overall system efficiency.

1.3 Objectives

The objectives of the Medicine Prescription Management System (MPMS) project are directly aligned with resolving the identified problems by developing five highly functional and integrated modules. The specific objectives are:

- To develop a Medicine Inventory Management Module that accurately tracks medicine details, quantities, expiry dates, and supplier information.

- To design a Prescription Management Module that allows pharmacists to create, edit, and manage prescriptions efficiently within the system.

- To build a Patient Information Management Module that securely stores and updates patient profiles and medical records.

- To implement a Sales and Billing Module that automates invoice generation and transaction recording for efficient sales management.

- To create a Reports and Analytics Module that generates comprehensive reports on stock, sales, prescriptions, and system usage for better decision-making.

1.4 Scope of the Project

1.4.1 Module to be developed

i. Medicine Inventory Management Module: Manages medicine details, stock levels, expiry dates, and suppliers.

ii. Prescription Management Module: Allows pharmacists to create and update prescriptions linked to patients and medicines.

iii. Patient Information Management Module: Stores and updates patient profiles and medical histories.

iv. Sales and Billing Module: Generates invoices and maintains sales records.

v. Reports and Analytics Module: Produces comprehensive reports on sales, stock, prescriptions, and system usage.

1.4.2 Target Users

The system is designed to serve and be utilized by three main groups:

1) Pharmacists
2) Patients
3) System Administrators

1.4.3 Technical Requirements

A critical technical requirement for this project is to demonstrate database interoperability. The system will be built to function with multiple database systems , specifically utilizing and showcasing compatibility across PostgreSQL, MySQL, and Microsoft SQL Server. The application development will be based on technologies including HTML, CSS, JavaScript, and PHP.

1.5 Project Significances

The Medicine Prescription Management System (MPMS) project holds significant importance on both operational and educational levels.

- Improved Operational Effectiveness.

  The initiative is primarily motivated by the necessity for precise and effective data management in pharmacies, particularly in settings where mistakes can directly impact patient well-being. The MPMS will minimize human mistakes, enhance healthcare information management, and improve the operational efficiency of pharmacies.

- Showcasing Database Interoperability.

  The project acts as a useful examination of how database systems can be created to function on various platforms, which is important information for database

administrators and developers. The project seeks to showcase effective database design and execution on various platforms by demonstrating compatibility with PostgreSQL, MySQL, and SQL Server.

- Practical Learning Experience.

  Through the adoption of this system, students will acquire practical experience with various Database Management Systems (DBMSs), database architecture, and actual system development. This aids in the enhancement of healthcare data management strategies.

- Conforming to Standards.

  The PPMS is designed to conform to contemporary digital healthcare management standards.

1.6 Conclusion

Chapter 1 successfully introduces the Medicine Prescription Management System (MPMS), defining it as a database-oriented application focused on modernizing pharmacy operations. The chapter clearly establishes the critical problems in current manual or disconnected systems, such as inventory errors and delayed communications. Furthermore, it sets forth five precise, module-based objectives to resolve these issues. Finally, the scope is clearly delineated, emphasizing the development of five core modules, serving three target user groups, and demonstrating essential database interoperability across PostgreSQL, MySQL, and SQL Server. This foundation solidifies the necessity and direction of the MPMS project.

CHAPTER 2: DATABASE SYSTEM METHODOLOGY AND PLANNING

2.1 Introduction

This chapter, Database System Methodology and Planning, serves as the blueprint for the systematic execution of the Medicine Prescription Management System (MPMS) project. Given the project's complexity, which includes developing five core functional modules and, critically, ensuring database interoperability across multiple platforms (PostgreSQL, MySQL, and SQL Server), a rigorous methodology and detailed planning are paramount to success. This section will first detail the chosen Database Development Methodology, justifying why a structured approach is necessary for maintaining data integrity and system reliability, while an iterative component ensures timely delivery and testing of the application layer. Subsequently, it will outline the principles of Project Management and Requirement handling, including the breakdown of tasks, resource allocation, and strategy for managing both functional and non-functional requirements. Finally, a clear set of Project Milestones will be defined, establishing a timeline against which progress will be measured to ensure the MPMS is delivered successfully, on time, and strictly within the defined scope. This planning framework aims to mitigate risks, ensure resource optimization, and guide the team through the challenges inherent in multi-platform database application development.

2.2 Database Development Methodology

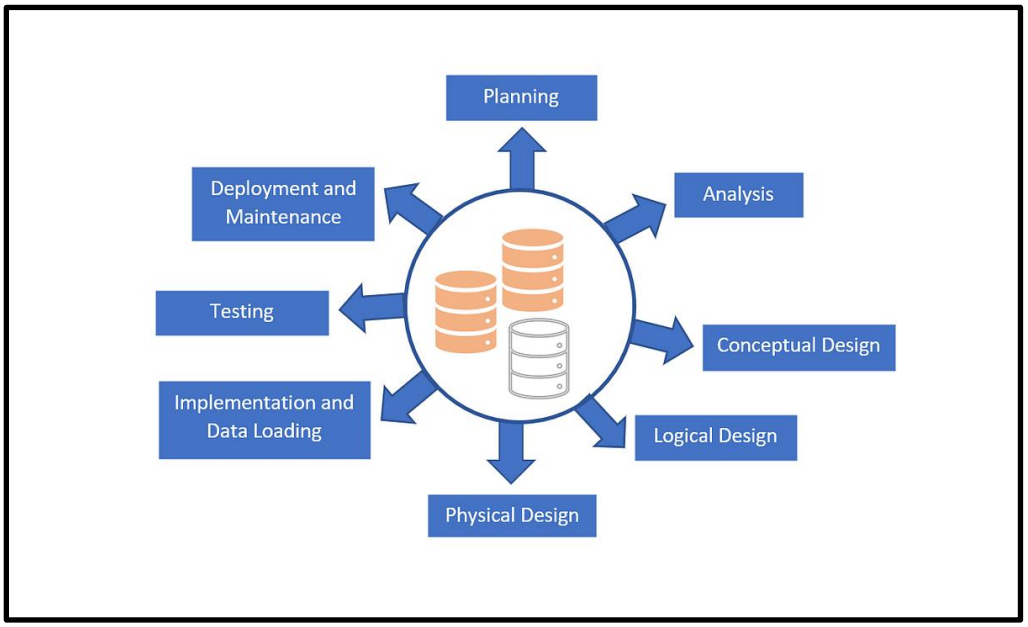


Figure 1: System Database Life Cycle (DBLC)

The Medicine Prescription Management System (MPMS) adopts the Database Life Cycle (DBLC) as the main methodology for database development in this project. The DBLC provides a structured approach to obtaining requirements, analysing data needs, designing the database, implementing it on multiple Database Management Systems (DBMSs), and maintaining it after deployment. The life cycle moves to the next stage only after the previous one has been completed and validated, which helps ensure that the MPMS database is consistent, reliable, and aligned with the system objectives.

The DBLC used in this project consists of six main stages: Database Initial Study, Database Design, Implementation and Loading, Testing and Evaluation, Operation, and Maintenance and Evolution. Each stage is described below in the context of the MPMS.

i. Database Initial Study

In this phase, the mission and objectives of the MPMS database are defined. The scope and boundaries of the project are identified by analysing the current pharmacy environment, including manual processes for inventory tracking, prescription handling, patient record storage, and sales recording. This study helps to identify key entities such as medicines, patients, prescriptions, users, and sales, as well as the data structures and relationships needed to support them. The main problems and constraints, such as inventory errors, incomplete patient histories, slow billing, and the need for interoperability across PostgreSQL, MySQL, and SQL Server, are also documented in this phase.

ii. Database Design

The second stage focuses on designing the MPMS database model to support secure and efficient pharmacy operations. This phase covers three levels of design:

- Conceptual design – development of the Entity-Relationship Diagram (ERD), describing entities (e.g. MEDICINE, PATIENT, PRESCRIPTION), their attributes, and relationships in a technology-independent form.
- Logical design – transformation of the ERD into normalized tables with clearly defined primary keys, foreign keys, and constraints, forming the data dictionary for PPMS.
- Physical design – definition of the actual storage structures and Data Definition Language (DDL) statements for each target DBMS (PostgreSQL, MySQL, SQL Server), including data types, indexes, and integrity constraints.

During this stage, the choice and configuration of DBMS software are also planned to ensure that the same logical model can be deployed consistently across multiple platforms.

iii. Implementation and Loading

This phase is the physical realization of the MPMS database design. The logical schema and DDL scripts are used to create the database objects—such as tables, views, indexes, stored procedures, and triggers—inside each selected DBMS. After the schemas are created, initial and sample data for medicines, patients, prescriptions, users, and sales are inserted into the tables to support development and testing activities. Application-level components (e.g. forms and modules for inventory, prescriptions, and billing) are then implemented to interact with the database using appropriate query languages and database drivers.

iv. Testing and Evaluation

Testing and evaluation ensure that the MPMS database and related application components function according to the specified requirements. This includes:

- Verifying that all tables, relationships, and constraints are created correctly and enforce data integrity.
- Testing stored procedures, triggers, and views to ensure that business rules—such as stock updates after sales or validation of prescription data—are executed correctly.
- Running queries and transactions from the PPMS interface to confirm that CRUD (Create, Read, Update, Delete) operations behave as expected.
- Evaluating performance and compatibility across PostgreSQL, MySQL, and SQL Server by executing equivalent operations and comparing the results.

Any issues identified during this stage lead to adjustments in the design or implementation before moving to the next stage.

v. Operation

Once the MPMS database has successfully passed testing, it is considered ready for operational use. In this stage, the database, the MPMS application, and its users (pharmacists, administrators, and patients) form a complete information system. Real pharmacy workflows are carried out using the system, including recording prescriptions, updating stock, processing sales, and generating reports. Operational data flows—such as daily sales transactions, prescription records, and inventory movements—are handled directly by the MPMS database in each deployed DBMS environment.

vi. Maintenance and Evolution

The final stage ensures that the PPMS database remains reliable, secure, and up-to-date over time. Regular maintenance tasks include:

- Preventive and corrective maintenance – performing backup and recovery, monitoring performance, and fixing data or schema issues when they arise.
- Adaptive maintenance – modifying the schema, queries, or stored procedures to accommodate new requirements, regulations, or changes in pharmacy workflows.
- User and access management – creating and updating user accounts, roles, and permissions for pharmacists, administrators, and other stakeholders.

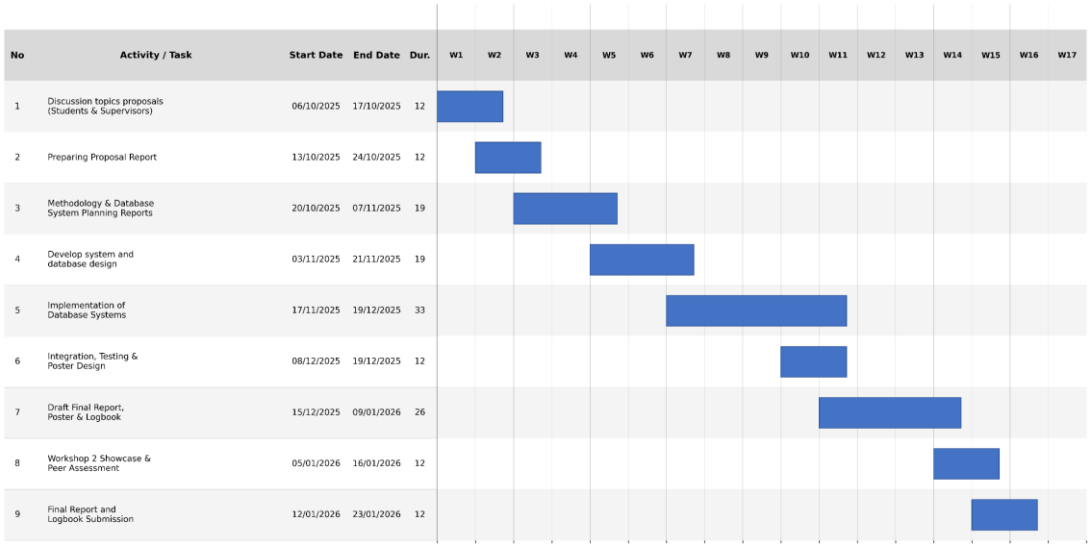
The database administrator is responsible for planning and executing these maintenance activities to preserve data integrity and support the continuous evolution of the MPMS as the pharmacy's needs grow.

2.3 Project Management and Requirement

Project management for the MPMS is based on breaking down the work into clear tasks aligned with the main modules and database activities, supported by simple scheduling tools

such as Gantt charts or task lists. Functional requirements, such as inventory tracking and prescription management, and non-functional requirements, such as performance, security, and interoperability, are documented and traced to ensure that each requirement is addressed by corresponding design and implementation tasks. Responsibilities are distributed among team members according to expertise, with regular discussions and reviews to monitor progress, address risks, and incorporate feedback from testing.

2.4 **Project Milestone**



No	Activity / Task	Start Date	End Date	Dur.
1	Discussion topics proposals (Students & Supervisors)	06/10/2025	17/10/2025	12
2	Preparing Proposal Report	13/10/2025	24/10/2025	12
3	Methodology & Database System Planning Reports	20/10/2025	07/11/2025	19
4	Develop system and database design	03/11/2025	21/11/2025	19
5	Implementation of Database Systems	17/11/2025	19/12/2025	33
6	Integration, Testing & Poster Design	08/12/2025	19/12/2025	12
7	Draft Final Report, Poster & Logbook	15/12/2025	09/01/2026	26
8	Workshop 2 Showcase & Peer Assessment	05/01/2026	16/01/2026	12
9	Final Report and Logbook Submission	12/01/2026	23/01/2026	12

Figure 2: Project Milestone 2

CHAPTER 3: DATABASE SYSTEM AND ANALYSIS

3.1 Introduction

This chapter focuses on the analysis of the existing pharmacy environment and the definition of the "to-be" system realized through the MPMS. It describes the current issues faced by pharmacies, explains the proposed system requirements, and provides visual representations of the system boundaries and data flows through the Context Diagram and Data Flow Diagram (DFD).

3.2 Current System Analysis

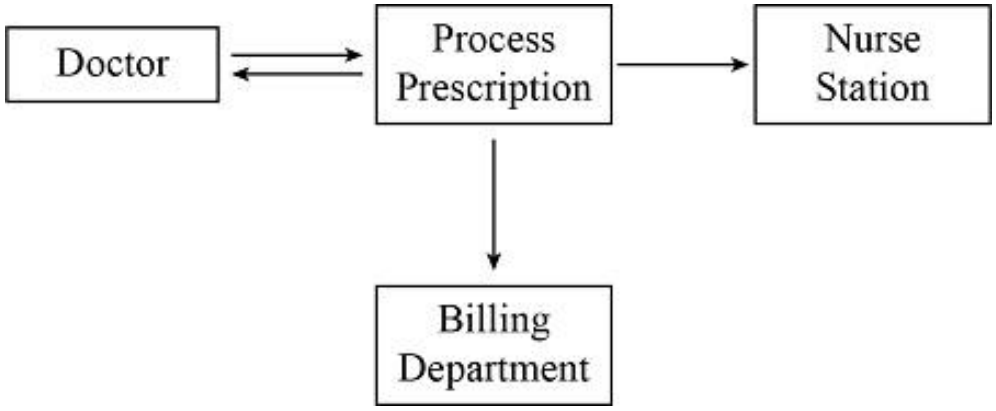


Figure 3: Project Target Users

3.3 System To-Be Analysis

3.3.1 Context Diagram

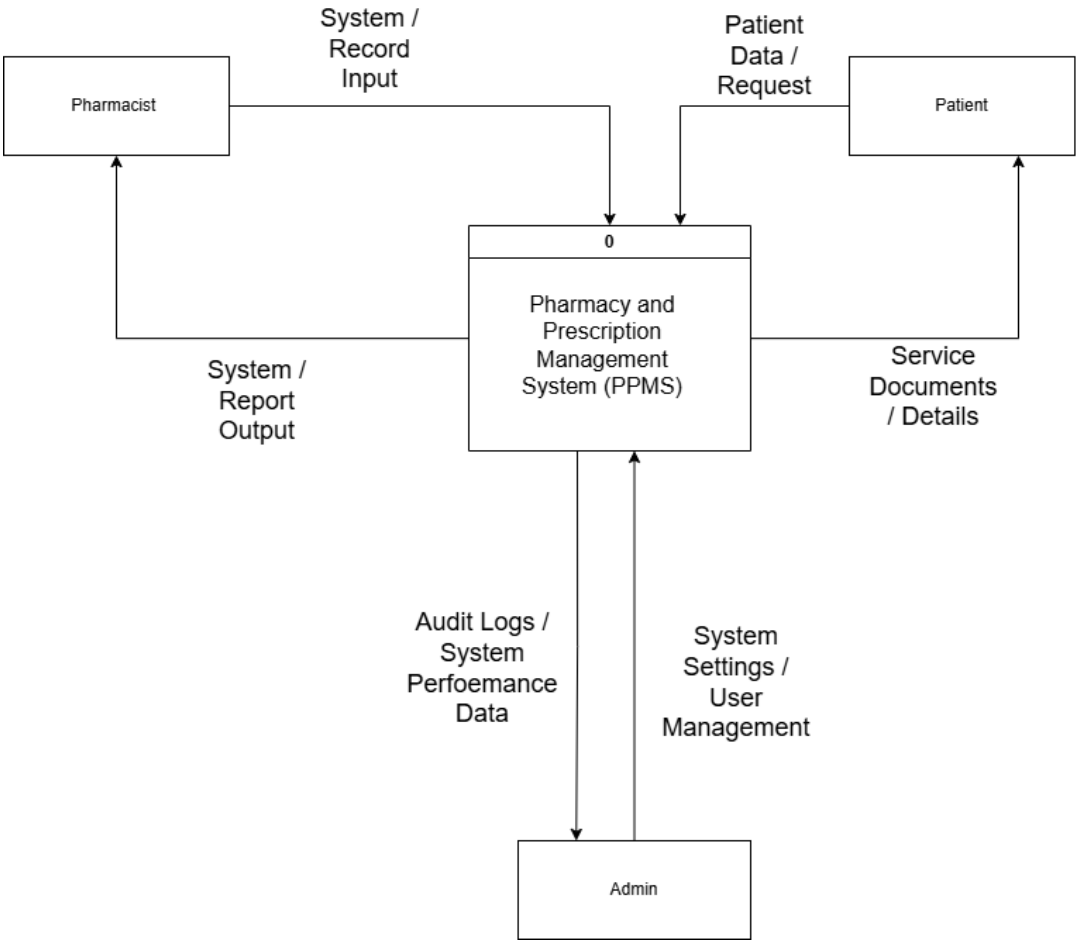


Figure 4: Context Diagram

3.3.2 Data Flow Diagram

Data Flow Diagram Level 1

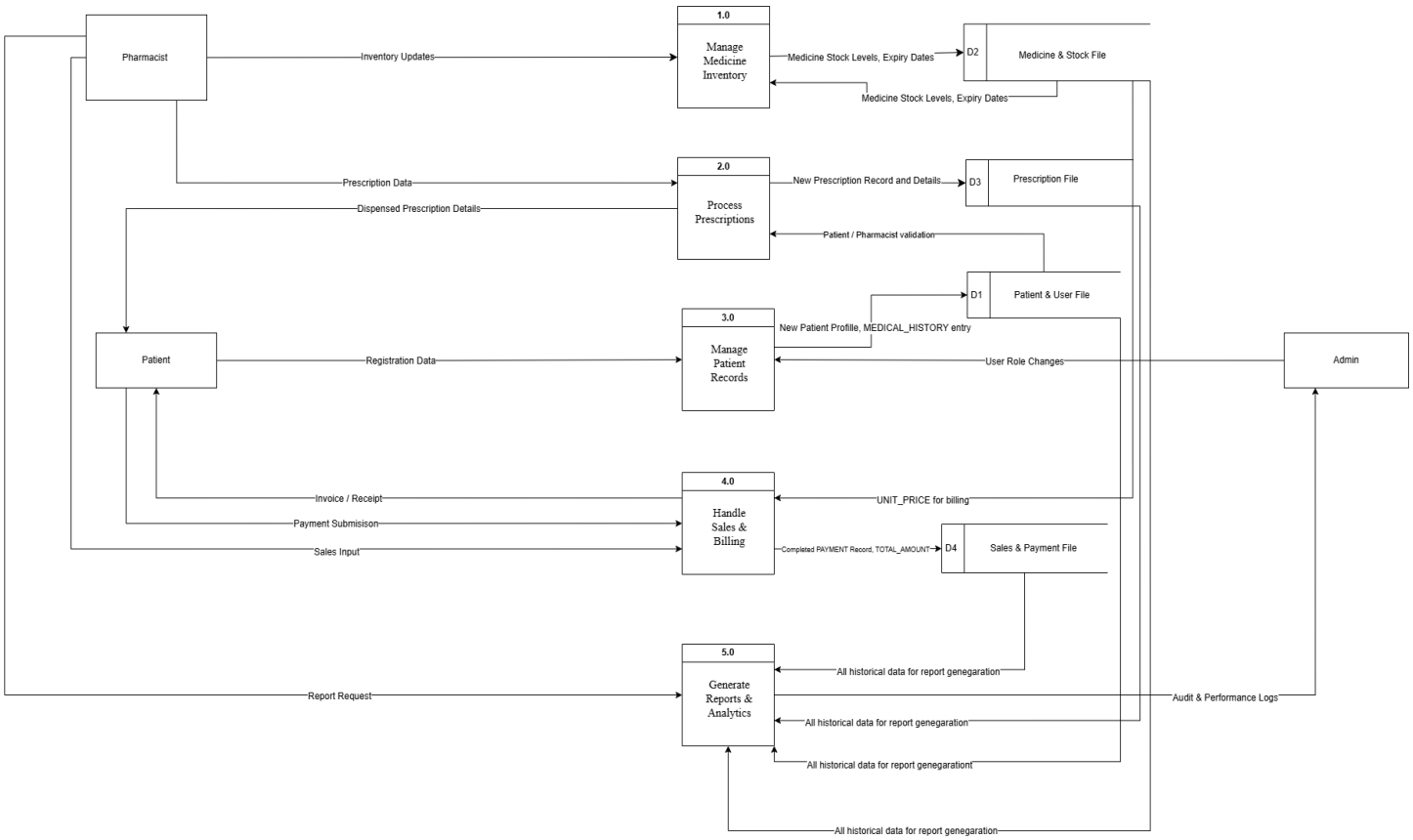


Figure 5: Data Flow Diagram Lvl 1 (DFD lvl 1)

3.4 Conclusion

In summary, Chapter 3 clarifies the shortcomings of the current manual or semi-manual systems and defines the requirements of the to-be MPMS through structured analysis. By using the Context Diagram and DFD, the chapter provides a clear view of system boundaries, major processes, data stores, and interactions, which forms a strong basis for the subsequent database design activities.

CHAPTER 4: DATABASE DESIGN

4.1 Introduction

This chapter documents the database design for the MPMS, covering conceptual, logical, and physical levels. The goal is to produce a robust, normalized schema that supports the functional requirements of the system while remaining compatible with PostgreSQL, MySQL, and SQL Server.

4.2 Conceptual Database Design

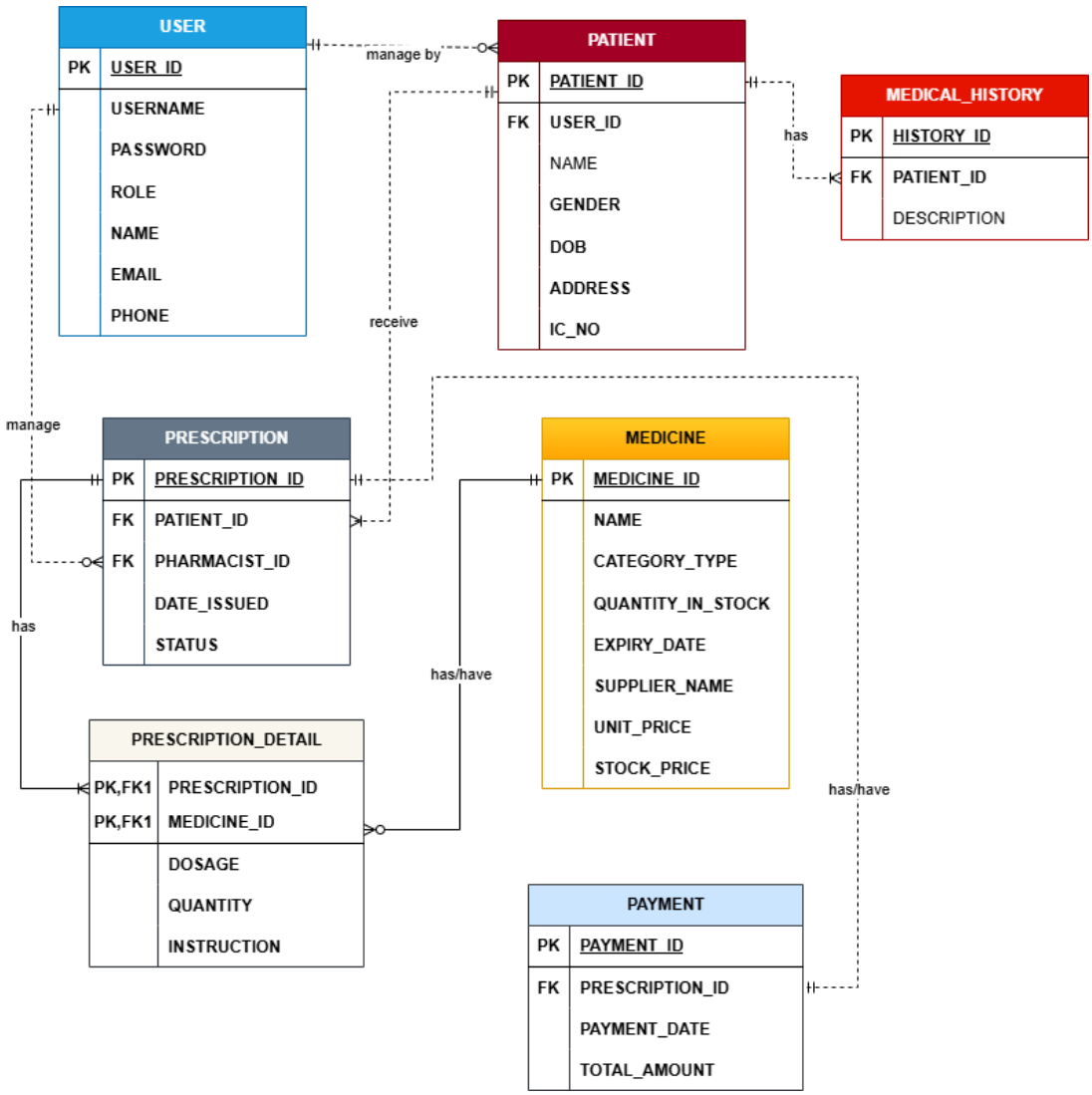


Figure 6 : ERD (Enhance Relationship Diagram)

4.3 Logical Database Design

4.3.1 Data Dictionary

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression
123 USER_ID	1	int	[v]	[v]	PRI		auto_increment	
A-Z USERNAME	2	varchar(50)	[v]	[ ]	UNI			
A-Z PASSWORD	3	varchar(255)	[v]	[ ]				
A-Z ROLE	4	varchar(20)	[v]	[ ]				
A-Z NAME	5	varchar(100)	[ ]	[ ]				
A-Z EMAIL	6	varchar(100)	[ ]	[ ]				
A-Z PHONE	7	varchar(20)	[ ]	[ ]				

Figure 7: User Table

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression
123 MEDICINE_ID	1	int	[v]	[v]	PRI		auto_increment	
A-Z NAME	2	varchar(100)	[v]	[ ]				
A-Z CATEGORY_TYPE	3	varchar(50)	[ ]	[ ]				
123 QUANTITY_IN_STOCK	4	int	[ ]	[ ]				
EXPIRY_DATE	5	date	[ ]	[ ]				
A-Z SUPPLIER_NAME	6	varchar(100)	[ ]	[ ]				
123 UNIT_PRICE	7	decimal(10,2)	[ ]	[ ]				
123 STOCK_PRICE	8	decimal(10,2)	[ ]	[ ]				

Figure 8: Medicine Table

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression
123 PATIENT_ID	1	int	[v]	[v]	PRI		auto_increment	
123 USER_ID	2	int	[ ]	[ ]	MUL			
A-Z GENDER	3	varchar(10)	[ ]	[ ]				
DOB	4	date	[ ]	[ ]				
A-Z ADDRESS	5	varchar(255)	[ ]	[ ]				
A-Z IC_NO	6	varchar(20)	[ ]	[ ]				
A-Z NAME	7	varchar(255)	[ ]	[ ]				

Figure 9: Patient Table

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression
123 PAYMENT_ID	1	int	[v]	[v]	PRI		auto_increment	
123 PRESCRIPTION_ID	2	int	[v]	[ ]	MUL			
PAYMENT_DATE	3	date	[ ]	[ ]				
123 TOTAL_AMOUNT	4	decimal(10,2)	[ ]	[ ]				

Figure 10: Payment Table

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression
123 PRESCRIPTION_ID	1	int	[v]	[v]	PRI		auto_increment	
123 PATIENT_ID	2	int	[v]	[ ]	MUL			
123 PHARMACIST_ID	3	int	[v]	[ ]	MUL			
DATE_ISSUED	4	date	[v]	[ ]				
A-Z STATUS	5	varchar(20)	[ ]	[ ]				

Figure 11: Prescription Table

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression
123 PRESCRIPTION_ID	1	int	[v]	[ ]	PRI			
123 MEDICINE_ID	2	int	[v]	[ ]	PRI			
A-Z DOSAGE	3	varchar(100)	[ ]	[ ]				
123 QUANTITY	4	int	[ ]	[ ]				
A-Z INSTRUCTION	5	varchar(255)	[ ]	[ ]				

Figure 12: Prescription Details Table

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression
123 HISTORY_ID	1	int	[v]	[v]	PRI		auto_increment	
123 PATIENT_ID	2	int	[v]	[ ]	MUL			
A-Z DESCRIPTION	3	text	[ ]	[ ]				

Figure 13: Medical History Table

4.4 Physical Database Design

4.4.1 Data Definition Language

User table

```
-- Table: public.USER

-- DROP TABLE IF EXISTS public."USER";

CREATE TABLE IF NOT EXISTS public."USER"

(

    user_id integer NOT NULL DEFAULT nextval('"USER_user_id_seq"'::regclass),

    username character varying(50) COLLATE pg_catalog."default" NOT NULL,

    password character varying(255) COLLATE pg_catalog."default" NOT NULL,

    role character varying(20) COLLATE pg_catalog."default" NOT NULL,

    name character varying(100) COLLATE pg_catalog."default",

    email character varying(100) COLLATE pg_catalog."default",

    phone character varying(20) COLLATE pg_catalog."default",
```

```
    CONSTRAINT "USER_pkey" PRIMARY KEY (user_id),
    CONSTRAINT "USER_username_key" UNIQUE (username)
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."USER"
    OWNER to postgres;
```

Payment Table

```
-- Table: public.payment
-- DROP TABLE IF EXISTS public.payment;
CREATE TABLE IF NOT EXISTS public.payment
(
  payment_id integer NOT NULL DEFAULT
nextval('payment_payment_id_seq'::regclass),
    prescription_id integer NOT NULL,
    payment_date date,
    total_amount numeric(10,2),
    CONSTRAINT payment_pkey PRIMARY KEY (payment_id),
    CONSTRAINT payment_prescription_id_fkey FOREIGN KEY
(prescription_id)
        REFERENCES public.prescription (prescription_id) MATCH SIMPLE
        ON UPDATE NO ACTION
       ON DELETE NO ACTION
)
TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public.payment
OWNER to postgres;
```

Prescription table

```
-- Table: public.prescription
-- DROP TABLE IF EXISTS public.prescription;
CREATE TABLE IF NOT EXISTS public.prescription
(
    prescription_id integer NOT NULL DEFAULT nextval('prescription_prescription_id_seq'::regclass),
    patient_id integer NOT NULL,
    pharmacist_id integer NOT NULL,
    date_issued date NOT NULL,
    status character varying(20) COLLATE pg_catalog."default",
    CONSTRAINT prescription_pkey PRIMARY KEY (prescription_id),
    CONSTRAINT prescription_patient_id_fkey FOREIGN KEY (patient_id)
        REFERENCES public.patient (patient_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT prescription_pharmacist_id_fkey FOREIGN KEY (pharmacist_id)
        REFERENCES public."USER" (user_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
```

```
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public.prescription
```

Medicine table

```
-- Table: public.medicine
-- DROP TABLE IF EXISTS public.medicine;
CREATE TABLE IF NOT EXISTS public.medicine
(
    medicine_id integer NOT NULL DEFAULT nextval('medicine_medicine_id_seq'::regclass),
    name character varying(100) COLLATE pg_catalog."default" NOT NULL,
    category_type character varying(50) COLLATE pg_catalog."default",
    quantity_in_stock integer,
    expiry_date date,
    supplier_name character varying(100) COLLATE pg_catalog."default",
    unit_price numeric(10,2),
    stock_price numeric(10,2),
    CONSTRAINT medicine_pkey PRIMARY KEY (medicine_id)
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public.medicine
```

Patient table

```
-- Table: public.patient

-- DROP TABLE IF EXISTS public.patient;

CREATE TABLE IF NOT EXISTS public.patient

(

    patient_id integer NOT NULL DEFAULT
nextval('patient_patient_id_seq'::regclass),

    user_id integer,

    gender character varying(10) COLLATE pg_catalog."default",

    dob date,

    address character varying(255) COLLATE pg_catalog."default",

    ic_no character varying(20) COLLATE pg_catalog."default",

    name character varying(255) COLLATE pg_catalog."default",

    CONSTRAINT patient_pkey PRIMARY KEY (patient_id),

    CONSTRAINT patient_user_id_fkey FOREIGN KEY (user_id)

        REFERENCES public."USER" (user_id) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.patient
```

Prescription detail table

```
-- Table: public.prescription_detail

-- DROP TABLE IF EXISTS public.prescription_detail
```

```
CREATE TABLE IF NOT EXISTS public.prescription_detail )
    prescription_id integer NOT NULL,
    medicine_id integer NOT NULL,
    dosage character varying(100) COLLATE pg_catalog."default",
    quantity integer,
    instruction character varying(255) COLLATE pg_catalog."default",
     CONSTRAINT prescription_detail_pkey PRIMARY KEY (prescription_id, medicine_id),
     CONSTRAINT prescription_detail_medicine_id_fkey FOREIGN KEY (medicine_id)
    REFERENCES public.medicine (medicine_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION,
  CONSTRAINT prescription_detail_prescription_id_fkey FOREIGN KEY (prescription_id)
    REFERENCES public.prescription (prescription_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION )
      TABLESPACE pg_default;
ALTER TABLE IF EXISTS public.prescription_detail;
```

4.5 Conclusion

In conclusion, this chapter has presented the complete database design of the Medicine Prescription Management System (MPMS), covering the conceptual, logical, and physical levels of database development. The physical database design was implemented using

structured Data Definition Language (DDL) statements to translate the logical schema into actual database tables within the PostgreSQL environment.

The defined tables including USER, patient, medicine, prescription, prescription_detail, and payment represent the core entities required to support pharmacy operations. Primary keys were applied to uniquely identify records, while foreign key constraints were used to enforce referential integrity between related tables, such as linking prescriptions to patients, pharmacists, and medicines. The use of appropriate data types and constraints ensures data consistency, accuracy, and reliability across the system.

Furthermore, the normalization of data into separate but related tables minimizes redundancy and improves database maintainability. The implementation of a composite primary key in the prescription_detail table effectively models the many-to-many relationship between prescriptions and medicines. Overall, the physical database design successfully supports the functional requirements of the MPMS and provides a stable foundation for application development, database integration, and testing, which are discussed in the subsequent chapters.

CHAPTER 5: INDIVIDUAL APPLICATION MODULES AND IMPLEMENTATION

5.1 Introduction

After obtaining the requirement and designing the component to show flow of the system, project implementation is where the project execution is taking place from the documented data into a real program. The focus of this chapter is to show the programming technique used to develop the system and the error handling. Programming technique is the technique used to describe the development process and the details of programming language code used.

5.2 Database Setup and Installation

The Medicine Prescription Management System (MPMS) was designed to support multiple Database Management Systems (DBMS), namely PostgreSQL, MySQL, and Microsoft SQL Server, in order to demonstrate database interoperability and portability.

- For PostgreSQL, the database was installed using the official PostgreSQL installer together with pgAdmin. A new database was created under the public schema, and tables were defined using PostgreSQL-compatible SQL scripts. Sequences were used to generate primary key values, and user access privileges were configured to allow secure application access.
- For MySQL, the MySQL Server was installed along with MySQL Workbench. A dedicated database schema was created, and equivalent tables were implemented using AUTO_INCREMENT for primary keys. User privileges were assigned to enable controlled access from the PPMS application.
- For Microsoft SQL Server, SQL Server Express was installed with SQL Server

Management Studio (SSMS). A new database was created, and tables were defined using Transact-SQL (T-SQL). IDENTITY columns were used to auto-generate primary keys, and constraints were applied to ensure data integrity.

5.3 Database Implementation

The MPMS database implementation applies database programming techniques to ensure data integrity and consistent business rule enforcement across PostgreSQL, MySQL, and Microsoft SQL Server. Core operations are handled at the database level using stored procedures, triggers, and constraints.

Stored procedures are used to standardize critical operations such as recording payment transactions. These procedures validate input data, ensuring that only valid prescription references and payment amounts are inserted into the database. Equivalent procedures were implemented using PL/pgSQL for PostgreSQL, MySQL stored procedure syntax, and Transact-SQL for SQL Server.

Triggers were implemented to automatically update medicine stock levels whenever prescription details are inserted. This mechanism ensures that inventory quantities are reduced in real time and prevents negative stock values across all supported DBMS platforms.

In addition, error handling is enforced through database constraints such as primary keys, foreign keys, UNIQUE, and NOT NULL restrictions, combined with validation logic within stored procedures and triggers. By implementing these techniques at the database level, the PPMS ensures reliable data processing, reduces application-level errors, and maintains

consistent system behavior across multiple database environments.

Example of error handling:

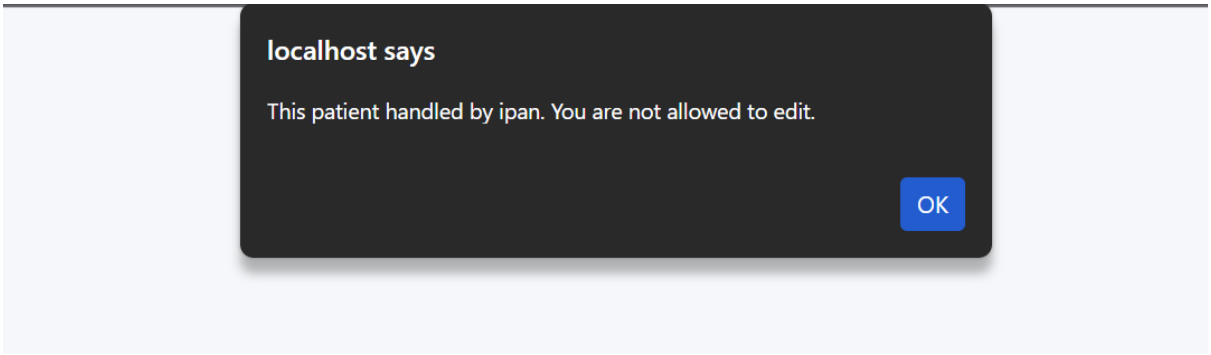


Figure 14: Error Handling

5.4 Application Modules

Module 1: User management module

The User Management Module is responsible for managing system users and controlling access to the Medicine Prescription Management System (MPMS). This module allows system administrators to add and delete user accounts for pharmacists, administrators, and patients. Each user is assigned a specific role that determines their level of access to system functions and data.

The module ensures system security by handling user authentication through login and logout functions. It also supports role-based access control, ensuring that sensitive operations such as modifying medicine records or accessing reports are restricted to authorized users only. By managing user credentials and permissions centrally, this module helps maintain data integrity and accountability within the system.

Module 2: Medicine inventory module

The Medicine Inventory Management Module manages all information related to medicines available in the pharmacy. This includes storing medicine details such as medicine name, category, dosage, price, quantity in stock, expiry date, and supplier information. The module allows pharmacists to add new medicines, update existing records, and remove expired or discontinued items.

This module plays a crucial role in preventing stock shortages and overstocking by enabling real-time monitoring of inventory levels. It also supports stock updates when medicines are sold or dispensed through prescriptions. By maintaining accurate and up-to-date inventory records, the module improves operational efficiency and reduces the risk of dispensing expired or unavailable medicines.

Module 3: Prescription module

The Prescription Management Module handles the creation, modification, and management of patient prescriptions. Pharmacists can record prescription details such as prescription date, prescribed medicines, dosage instructions, quantity, and the associated patient information. Each prescription is linked to both the patient and the medicines stored in the inventory database.

This module helps reduce prescription errors by ensuring that only valid medicines with sufficient stock can be prescribed. It also enables pharmacists to retrieve and review prescription histories for reference and auditing purposes. By centralizing prescription records, the module improves communication between healthcare providers and pharmacists while ensuring accurate and secure prescription handling.

Module 4: Sales and Billing module

The Sales and Billing Module is responsible for managing pharmacy sales transactions and generating invoices. It records sales details including sold medicines, quantities, total price, payment method, and transaction date. This module automatically calculates the total bill based on medicine prices and quantities selected.

In addition, the module updates medicine stock levels in real time after each successful transaction, ensuring inventory data remains accurate. By automating the billing process, this module reduces manual calculation errors, speeds up customer checkout, and provides reliable sales records for financial tracking and reporting.

Module 5: Reports and Analysis module

The Reports and Analysis Module provides analytical and reporting capabilities to support management decision-making. This module generates various reports such as medicine stock reports, sales summaries, prescription reports, and user activity reports. These reports can be filtered by date, medicine type, or transaction category.

The module helps pharmacy administrators monitor system performance, identify sales trends, and track fast-moving or low-stock medicines. By presenting structured and accurate information, this module enables better planning, improves inventory control, and supports strategic decision-making within the pharmacy.

5.5 User Interface

Login page

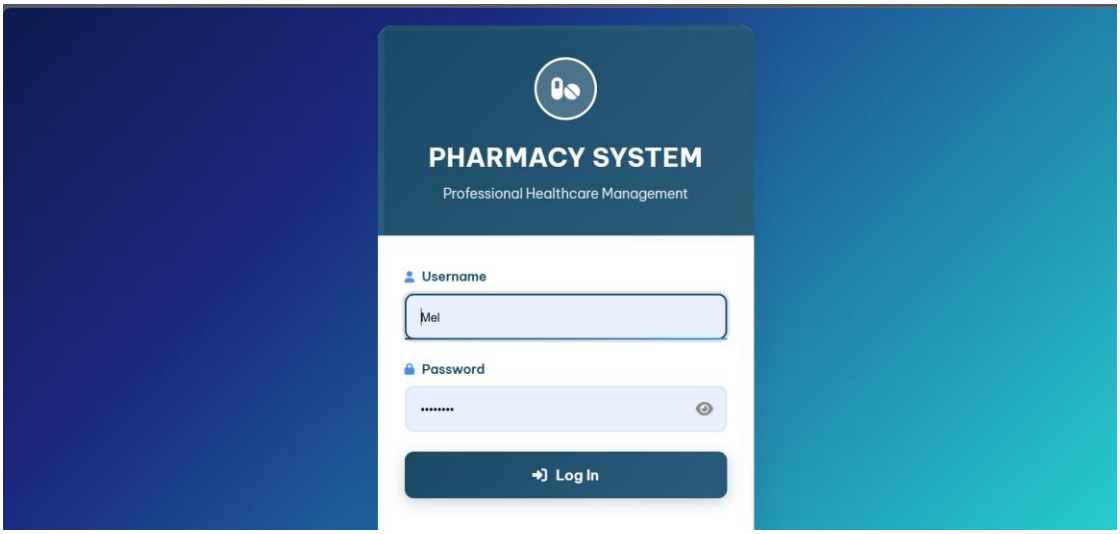


Figure 15: Login page 1

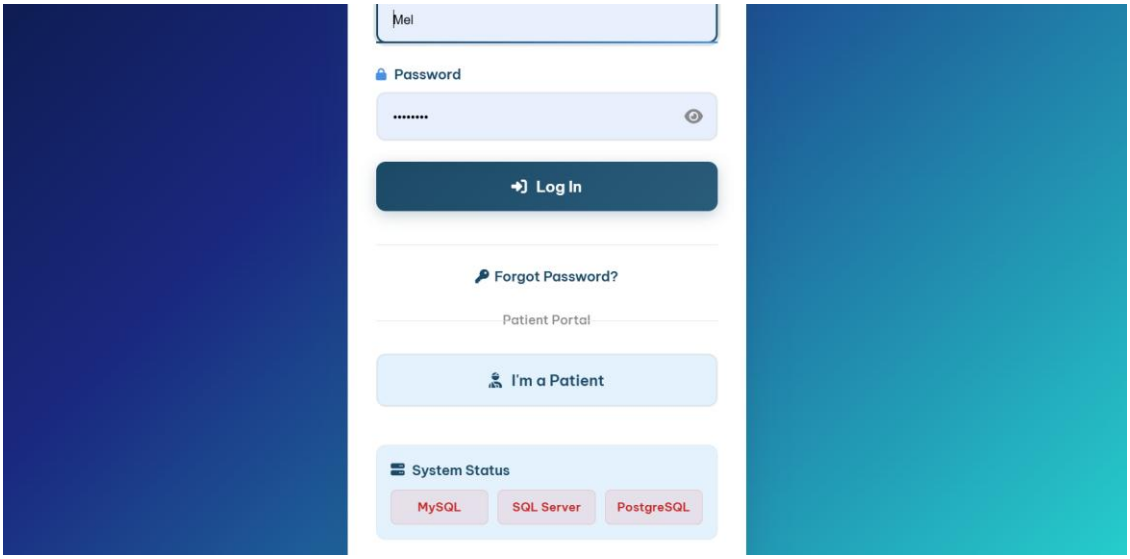


Figure 16: Login page 2

System Dashboard

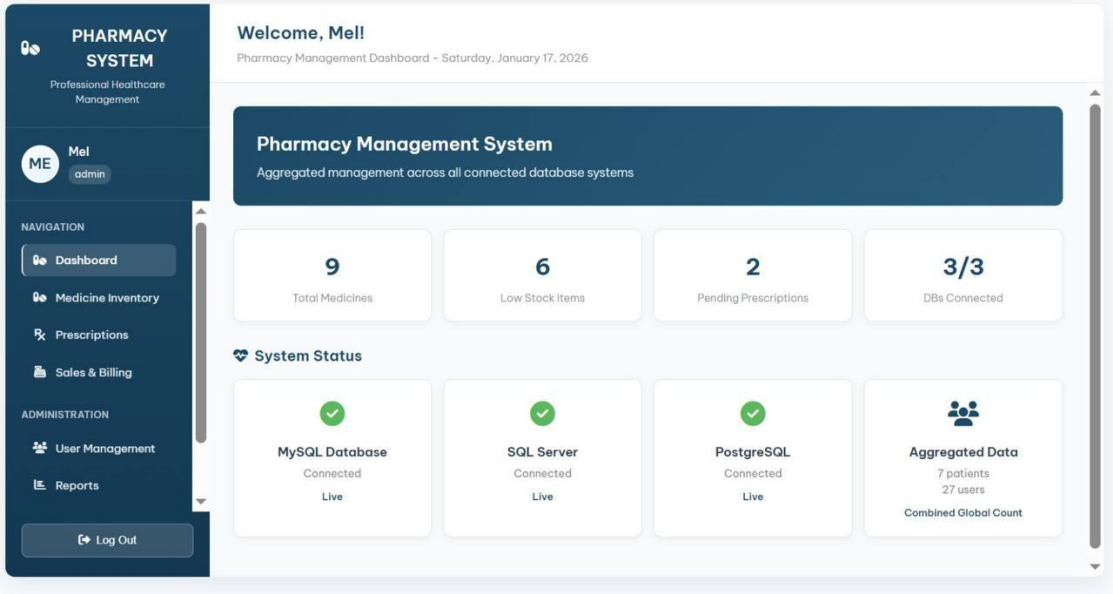


Figure 17: Dashboard

Medicine Inventory page

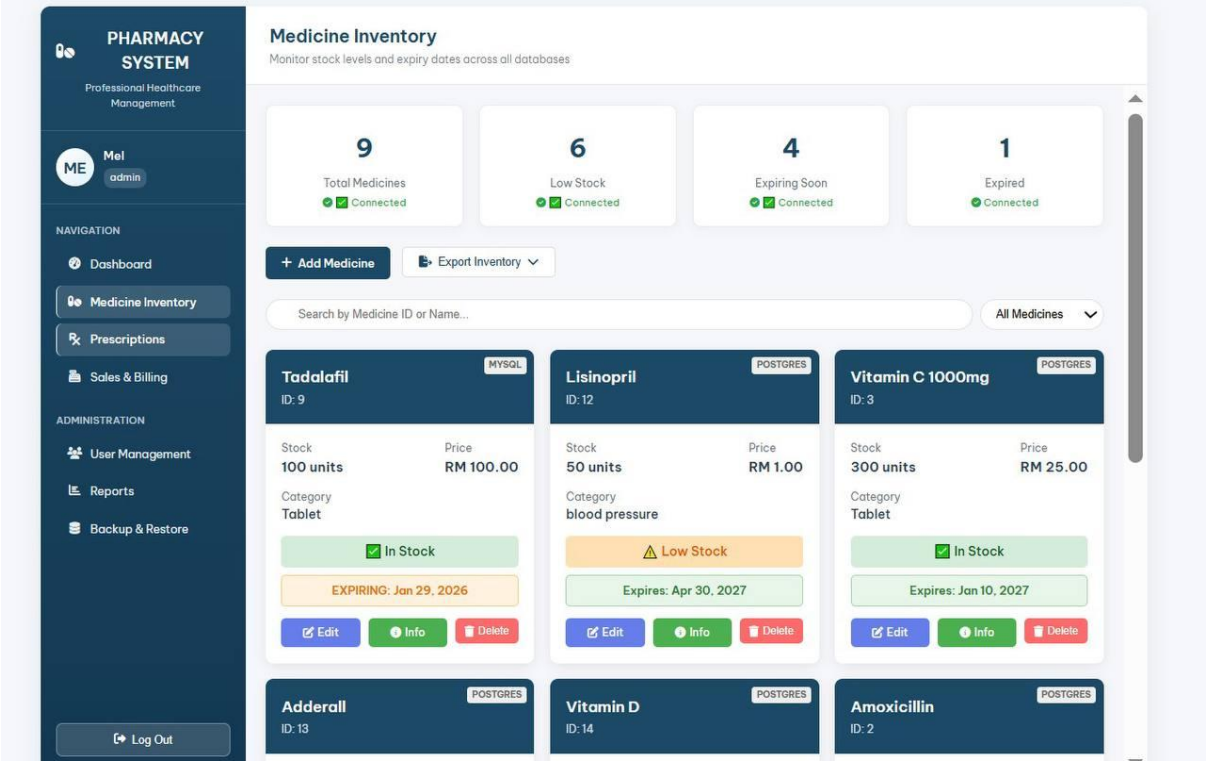


Figure 18: Medicine Inventory

Medicine inventory page (Add medicine)

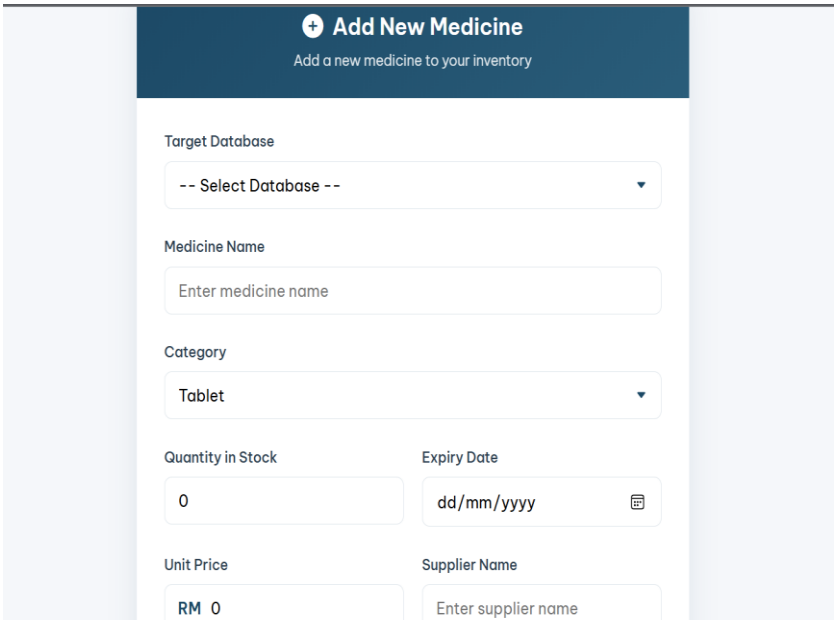


Figure 19: Add Medicine Page

Prescription page

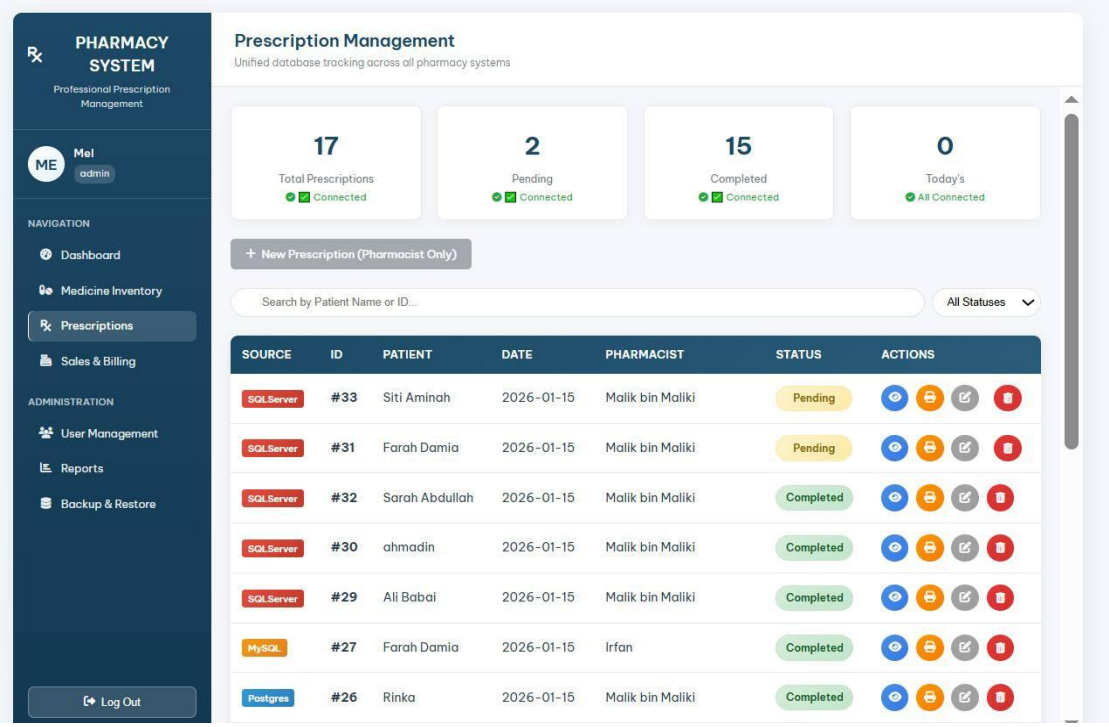


Figure 20: Prescription Page

Prescription page (view prescription record)

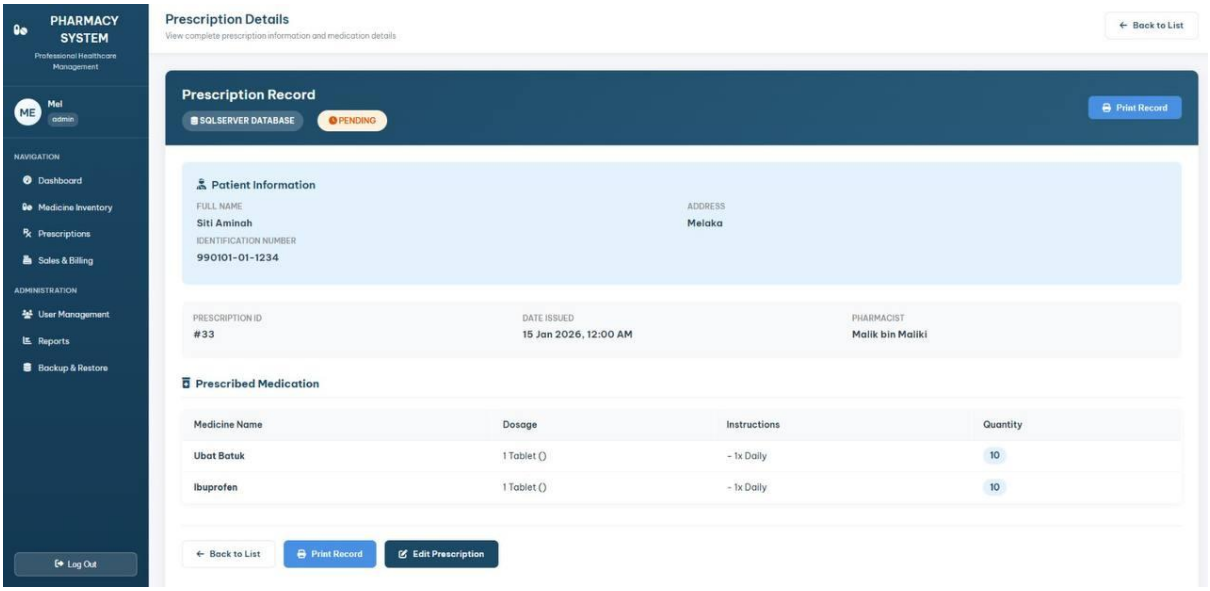


Figure 21: View Prescription Page

Sales and Billing page

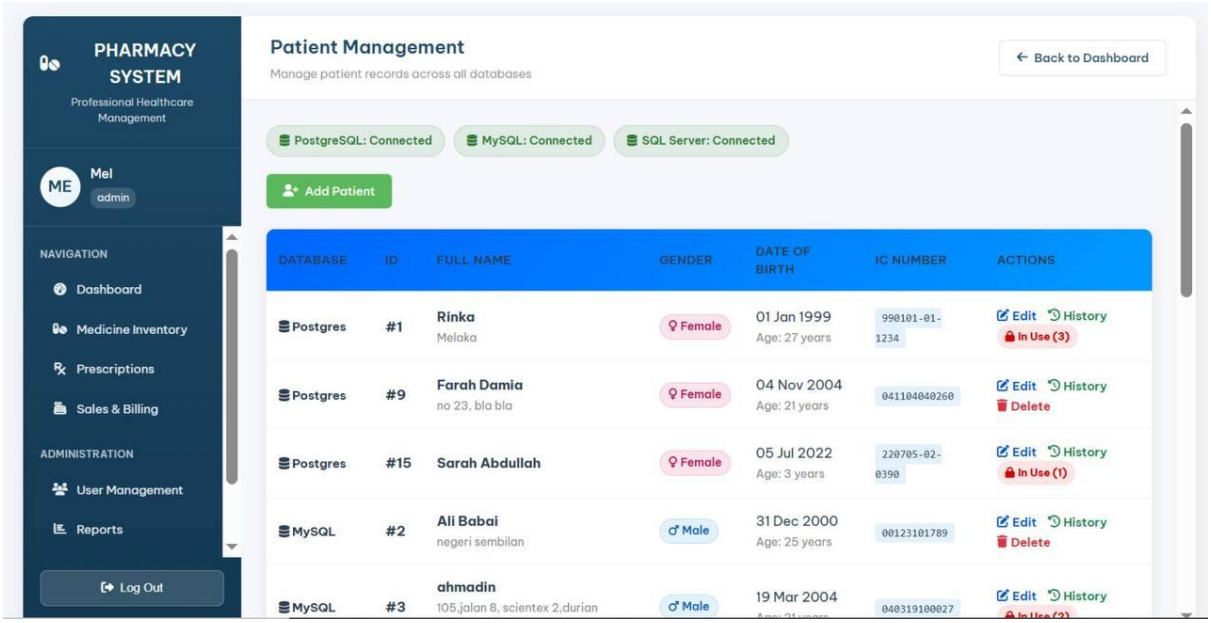


Figure 22: Sales and Billing Page

Sales and Billing page (receipt)

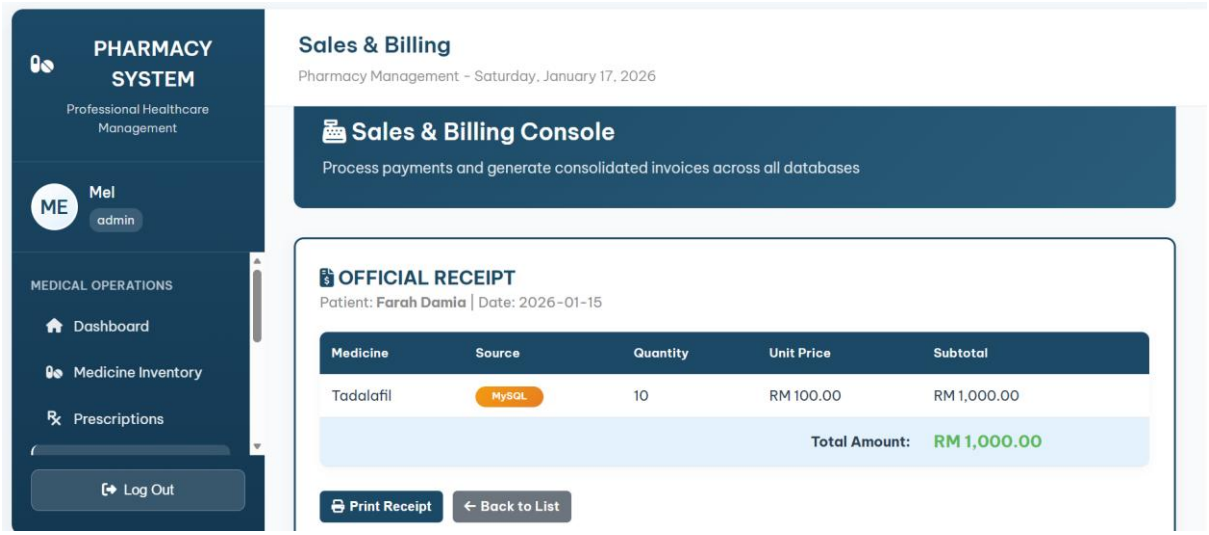


Figure 23: Receipt Page

User management page

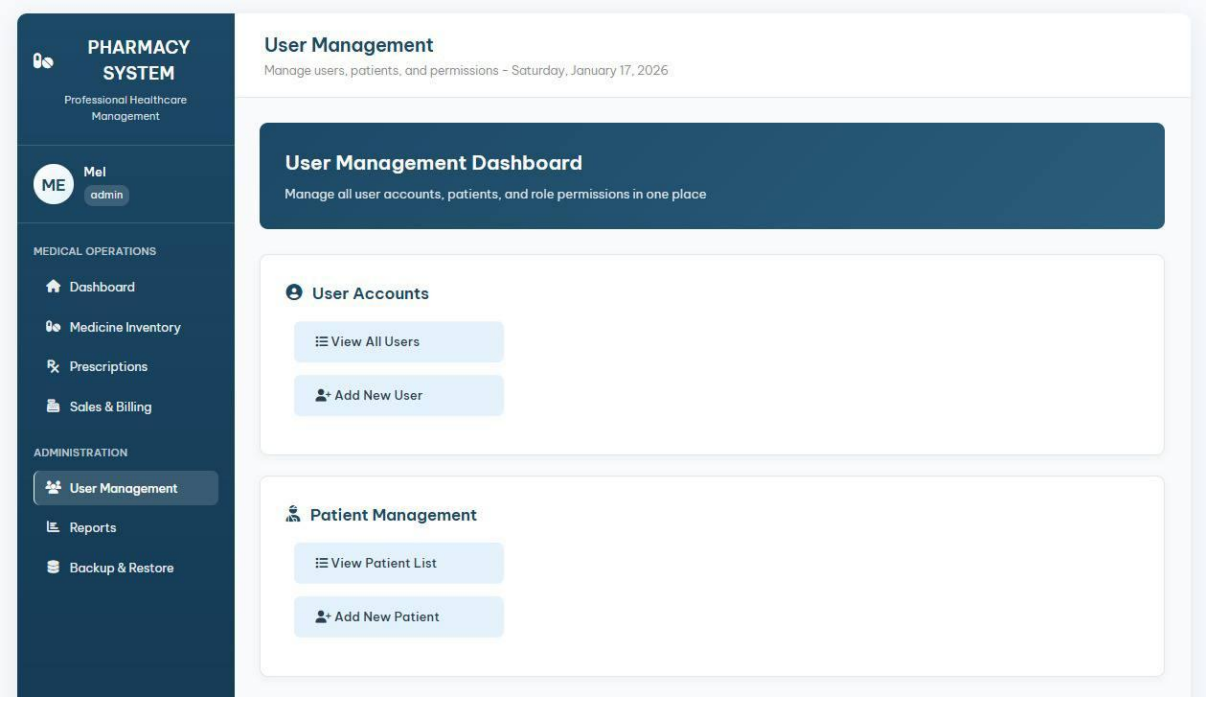


Figure 24: User Management Page

User management (user list)

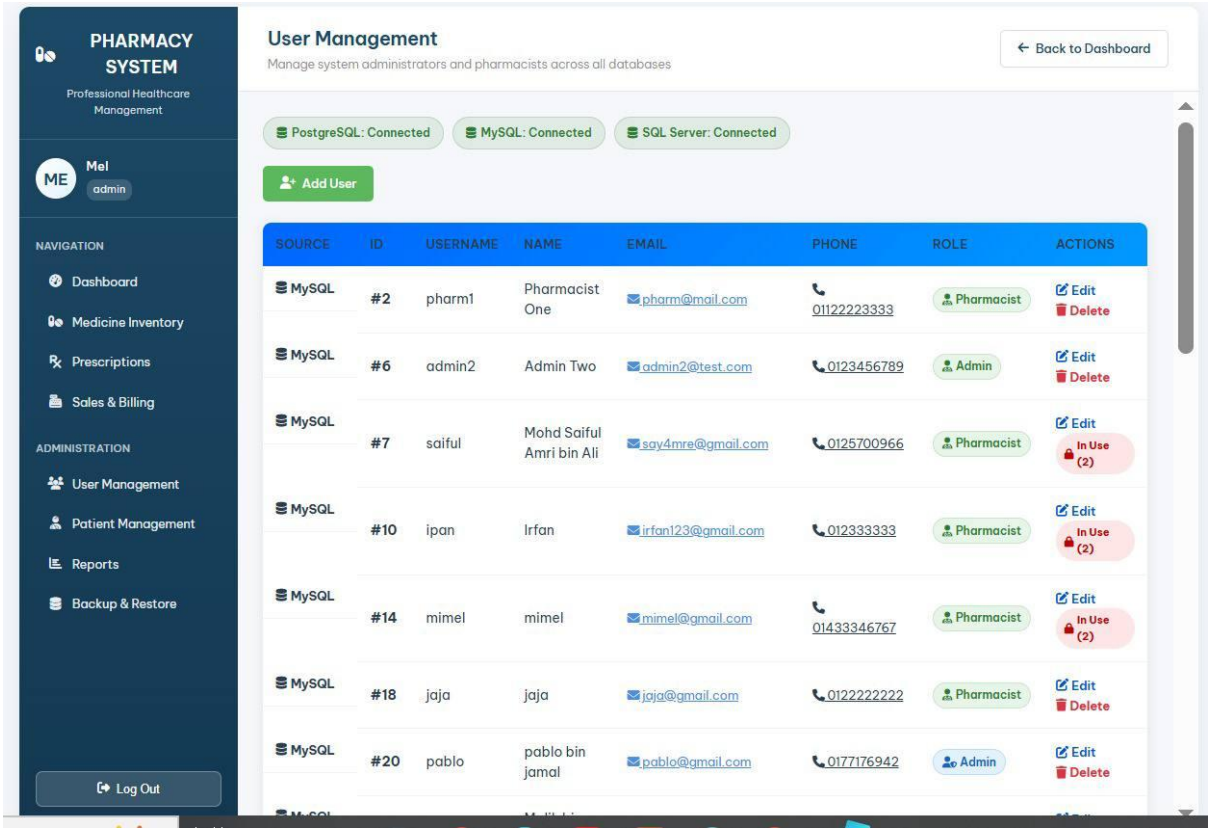


Figure 25: User List Page

User management (patient management)

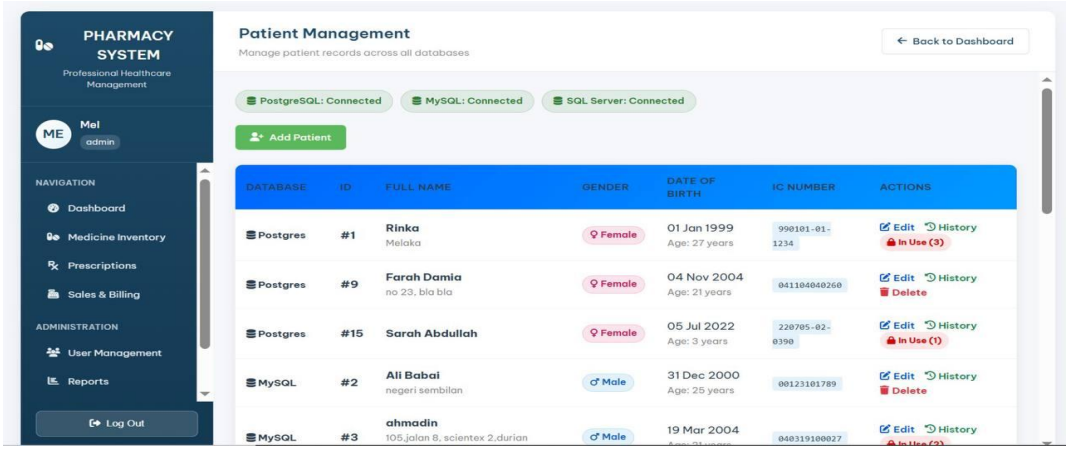


Figure 26: Patient Management Page

Reports and Analysis page

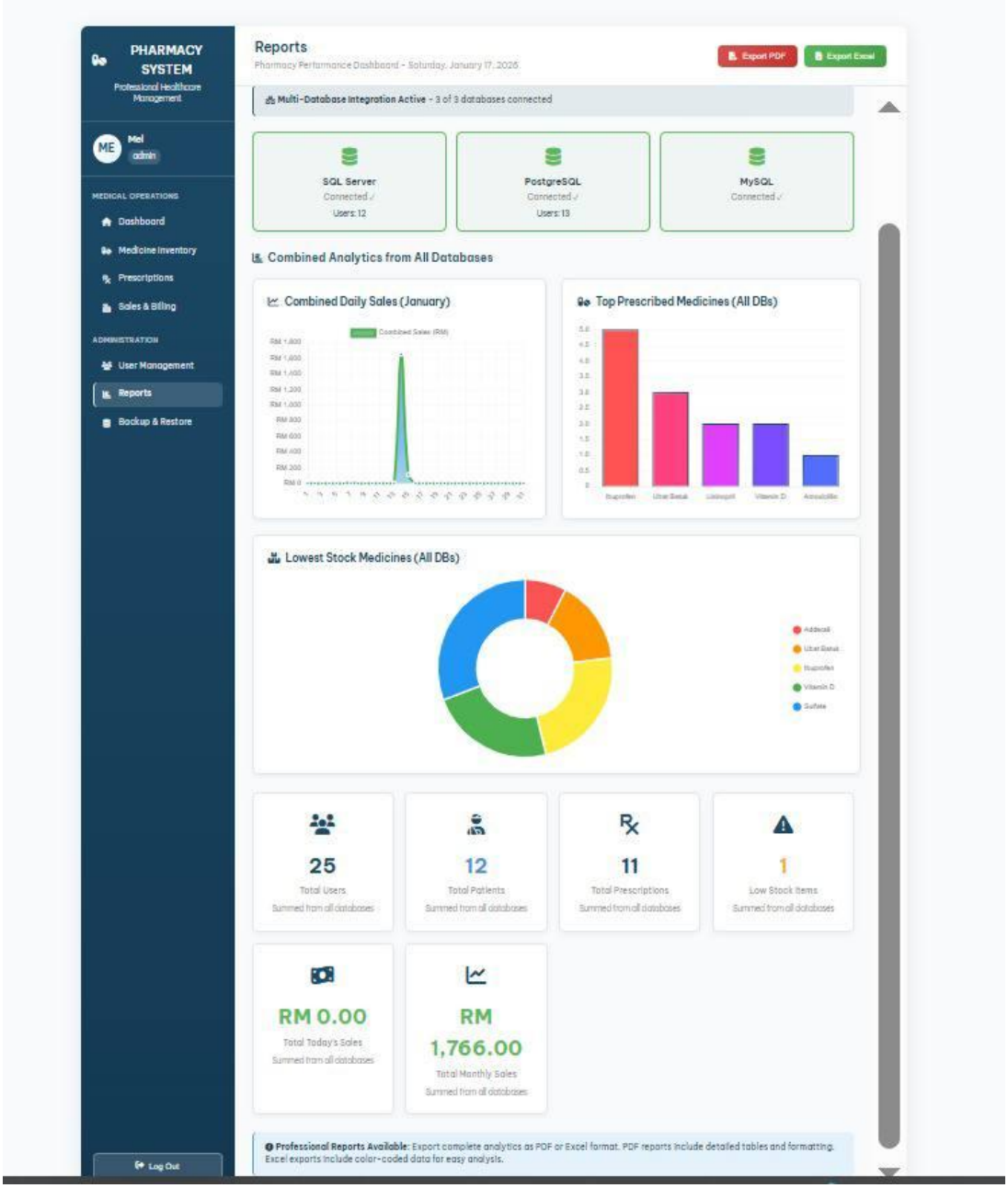


Figure 27: Report and Analysis Page

Backup & Restore page

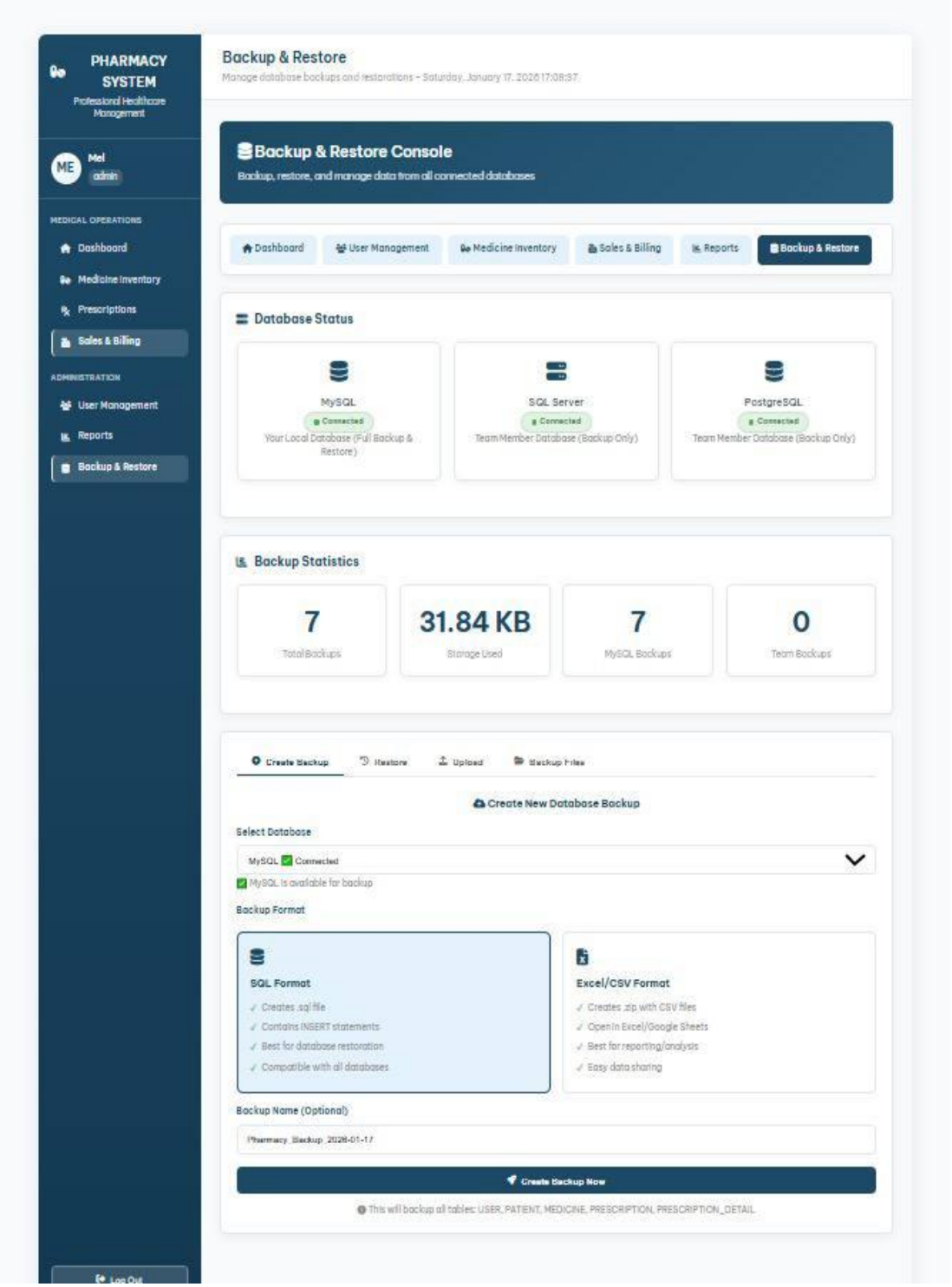


Figure 28: Backup and Restore Page

Patient page

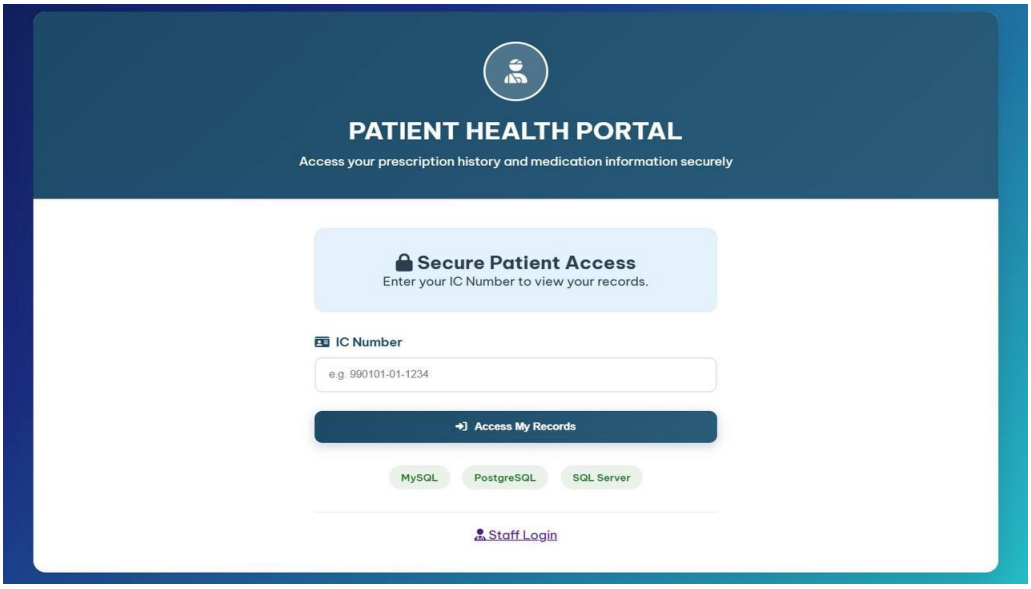


Figure 29: Patient Login Page

Patient view page

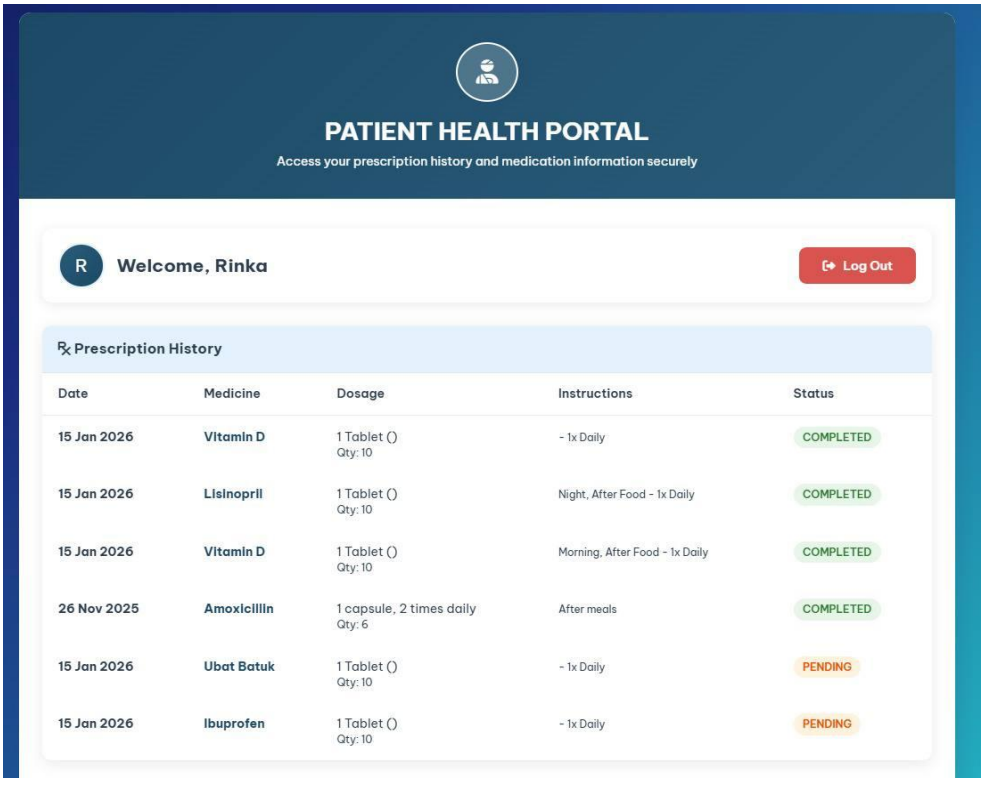


Figure 30: View Patient Prescription Page

5.6 Conclusion

In conclusion, this section has presented the implementation of the individual application modules including functionality of the Medicine Prescription Management System (MPMS). The User Management Module ensures secure access control and proper user authentication, while the Medicine Inventory Management Module enables accurate tracking of medicine records and stock levels. The Prescription Management Module supports efficient handling of patient prescriptions and reduces the risk of medication errors. In addition, the Sales and Billing Module automates transaction processing and ensures real-time updates to inventory data. Finally, the Reports and Analysis Module provides meaningful insights through structured reports, supporting effective monitoring and decision-making.

CHAPTER 6: DATABASE INTEGRATION AND TESTING

6.1 Introduction

This chapter explains how the Medicine Prescription Management System (MPMS) database is installed, integrated, and tested across the three supported Database Management Systems (DBMS): PostgreSQL, MySQL, and Microsoft SQL Server. Database integration in MPMS focuses on deploying the same logical schema on each platform and ensuring that all modules—inventory, prescriptions, patient records, sales, and reports—can access and manipulate data consistently regardless of the underlying DBMS. The chapter also describes the testing activities carried out to verify data integrity, business rules, and performance, including the execution of structured test cases on tables, constraints, stored procedures, and triggers. In addition, database management and administration tasks such as backup, restore, and user access control are documented to demonstrate how the MPMS database can be maintained in an operational environment.

6.2 Database Setup and Installation

The Medicine Prescription Management System (MPMS) database was deployed and configured using three Database Management Systems (DBMS), namely PostgreSQL, MySQL, and Microsoft SQL Server, to demonstrate database interoperability and portability. Each DBMS was installed on a local development environment and configured according to standard installation procedures.

For PostgreSQ**L**, the installation was carried out using the official PostgreSQL installer, which includes the pgAdmin management tool. During installation, a superuser account was

created, and a new database schema was defined under the public schema. Required tables were created using Data Definition Language (DDL) scripts, and database connections were configured for application access.

For MySQL, the MySQL Server package was installed. A database was created, and user privileges were assigned to allow full access to the MPMS schema. The same logical database structure was implemented using compatible SQL syntax to ensure consistency with the PostgreSQL version.

For SQL Server, SQL Server was installed. A new database was created, and tables were defined using Transact-SQL (T-SQL). Primary keys, foreign keys, and constraints were carefully configured to mirror the schema implemented in PostgreSQL and MySQL.

DBeaver was used as a cross-platform database management tool to support the setup and configuration of the MPMS databases. It enabled consistent connectivity to PostgreSQL, MySQL, and Microsoft SQL Server from a single interface, allowing database creation, schema deployment, and execution of SQL scripts. The use of DBeaver simplified database installation and verification by providing a unified environment for managing multiple DBMS platforms during development.

ZeroTier was also used as it was utilized to establish a secure virtual private network that enabled remote access to the MPMS database servers. This configuration allowed team members to connect to PostgreSQL, MySQL, and SQL Server instances across different locations as if they were on the same local network. The use of ZeroTier ensured reliable and secure database connectivity during development, integration, and testing activities.

6.3 Database Integration and Testing

Database integration in the MPMS focuses on ensuring that the application interacts correctly with PostgreSQL, MySQL, and SQL Server using the same logical data model. The application layer was configured to connect to each DBMS by modifying database connection parameters such as host, database name, username, and password.

Testing activities were carried out to verify database functionality and data integrity. These tests include:

- **CRUD Testing**: Insert, retrieve, update, and delete operations were performed on all major tables such as USER, patient, medicine, prescription, and payment to ensure correct data manipulation.

- **Referential Integrity Testing**: Foreign key constraints were tested by attempting to insert invalid references, ensuring that the database correctly prevents inconsistent data.

- **Transaction Testing**: Sales and prescription transactions were tested to confirm that related tables are updated correctly, including stock reduction after medicine dispensing.

- **Cross-DBMS Compatibility Testing**: Identical queries and operations were executed on PostgreSQL, MySQL, and SQL Server to ensure consistent behavior and results across all platforms.

6.4 Database Management and Administration

Database management and administration tasks were implemented to ensure data reliability, security, and recoverability. These tasks include user management, backup, and restore operations across all supported DBMS platforms.

The pg_dump utility in PostgreSQL was used to create logical backups, which allowed the complete database structure and data to be stored in a portable format and restored using the pg_restore command when needed. SQL-based backups of data and schema definitions were made for MySQL using the mysqldump utility. These backups can be readily restored by re-importing the backup files into the MySQL server. When it came to Microsoft SQL Server, entire database backups were made and saved using SQL Server Management Studio (SSMS), which was used for database administration operations.backup files and used the integrated database restore feature.

These administrative practices ensure that the MPMS database remains manageable, secure, and recoverable, supporting long-term system stability and maintenance across all supported DBMS platforms.

GitHub was used as a version control platform to manage database scripts, application source code, and project documentation throughout the development of the MPMS. It enabled collaborative development by tracking changes, maintaining version history, and supporting rollback when necessary. The use of GitHub contributes to effective database administration and long-term system maintenance by ensuring that all database-related changes are documented and controlled.

Basic administrative tasks such as user privilege assignment and password management were also carried out to ensure secure access and stable system performance. These management procedures ensure that the MPMS database remains maintainable and recoverable in an operational environment

6.5 Conclusion

In conclusion, this chapter has outlined the setup, integration, and testing of the MPMS database across PostgreSQL, MySQL, and Microsoft SQL Server. The database schema was successfully deployed on all three platforms, demonstrating database interoperability and consistent system behavior.

Database testing confirmed that core operations, referential integrity, and transactions function correctly across the different DBMS environments. Backup and restore procedures were also implemented to ensure data reliability and recoverability. Overall, the results show that the PPMS database is stable, portable, and ready to support the system's operational requirements.

CHAPTER 7: CONCLUSION

7.0 Introduction

In conclusion, the Medicine Prescription Management System (MPMS) has been developed to meet the objectives defined at the beginning of this project. Each module and database component was designed to follow proper database development standards while supporting safe and efficient pharmacy operations. Before adoption, the core functions of the system were tested to ensure that prescriptions, inventory, patient information, and sales can be processed correctly. Throughout this report, every aspect of the MPMS that ranges from problem analysis, database design, multi-DBMS implementation, up to integration and testing that has been discussed in detail so that users and stakeholders can clearly understand how the system operates. The challenges faced during development, especially related to interoperability across PostgreSQL, MySQL, and SQL Server, have led to valuable learning and a more robust solution. Ultimately, the PPMS is expected to help pharmacies manage medicines and prescriptions more accurately while saving time and effort for pharmacists and patients.

7.2 Achievement and Application in SDG

At the start of the project, several key modules were identified as necessary to solve the problems found in the existing pharmacy environment. These include the Medicine Inventory Management Module, Prescription Management Module, Patient Information Management Module, Sales and Billing Module, and Reports and Analytics Module. All of these modules have been successfully implemented and integrated with the underlying database architecture. The system is capable of running on three different DBMS platforms

like PostgreSQL, MySQL, and SQL Server and demonstrating the intended database interoperability. The objectives and scope defined in Chapter 1 have therefore been fulfilled, as the system can now handle inventory records, prescriptions, patient profiles, and sales transactions in a structured and reliable way. Functional testing carried out in the implementation and testing chapters shows that the main features work as intended within the defined project constraints.

7.3 Project Limitation

Although the MPMS achieves its core goals, several limitations remain. First, the current version focuses on a single-pharmacy environment and does not yet support multi-branch operations or real-time synchronization between outlets. Second, security features such as encryption of sensitive data, detailed audit trails, and advanced access control are kept at a basic level suitable for an academic project rather than a production healthcare system. Third, the reporting and analytics functions are limited to summary reports and do not yet offer advanced data visualisation or predictive analysis. In addition, while the system is compatible with three DBMS platforms, performance and stress testing under high transaction loads have been minimal, so scalability for large pharmacies has not been fully validated. These limitations provide clear directions for future enhancement of the MPMS.

7.4 Suggestion and Improvement

Several improvements can be implemented in future iterations of the MPMS. The system can be extended to support multi-branch pharmacies with centralized stock control and consolidated reporting. Security can be strengthened by introducing role-based access control, password policies, data encryption, and more comprehensive logging of user activities. The Reports and Analytics Module can be upgraded with interactive dashboards,

trend analysis, and forecasting features to assist managers in making data-driven decisions. Integration with external systems, such as e-prescription platforms, barcode scanners, or online payment gateways, would also make the system more practical in real pharmacy settings. Finally, developing a responsive web or mobile interface would allow pharmacists and patients to access key functions more conveniently, further increasing the usability and impact of the MPMS.

7.5 Potential Commercialization

The current implementation of the MPMS is primarily designed as an academic project, but it has clear potential for real-world use, especially in small and medium-sized community pharmacies. With its modular structure and ability to run on widely used DBMS platforms, the system could be packaged as an affordable solution for pharmacies that currently rely on manual records or simple spreadsheets. To be fully commercialised, additional work would be required to harden security, comply with healthcare regulations, improve user experience, and provide comprehensive technical support. If these enhancements are made and the system is tested extensively in real operational environments, the MPMS could evolve into a practical product that supports accurate medicine management, smoother prescription handling, and better decision-making for pharmacy owners.

7.6 Conclusion

Pharmacy and prescription management systems play a critical role in ensuring that medicines are dispensed safely and that pharmacy operations run smoothly. A poorly designed system can lead to inventory errors, delayed treatments, and difficulties in tracking patient records, which may negatively affect both patients and pharmacists. Through this project, the MPMS has addressed key issues identified in the problem statements by

providing structured modules for inventory, prescriptions, patient information, sales, and reporting, all supported by a well-designed database that can operate on multiple DBMS platforms. The planned modules have been implemented, the main objectives and scope have been achieved, and the system has undergone functional and integration testing to minimise operational problems. Overall, the project can be considered a success, providing both a working prototype for pharmacy operations and a strong foundation for future enhancement and possible commercial deployment.