

Philldesk
Online Pharmacy Management System

K R K Dalpatadu

June 2025



Philldesk

Online Pharmacy Management System

Name:K.R.K.Dalpatadu

Index no: 2370271

Registration No: R237027

Name of the Supervisor

Mr. R.M. Senaka Bandara Tilakarathne


June 2025



**This dissertation is submitted in partial fulfilment of the requirement of the Degree of
Bachelor of Information Technology (External) of the
University of Colombo School of Computing**

Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any University/Institute, and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text.

Signature of Candidate: 

Date: 30/06/2025.

Name of Candidate: K.R.K. Dalpatadu

Countersigned by:

Signature of Supervisor(s)/Advisor(s): .. 

Date: 30/06/2025

Name(s) of Supervisor(s)/Advisor(s): R.M. Senaka Bandara Tilakaratne

Abstract

PillDesk is a secure and efficient online pharmacy management system made to address common challenges faced by retail pharmacies. This system aims to automate core pharmacy operations such as inventory management, prescription handling, billing, and reporting. The scope includes features like role-based access control, real-time notifications for stock and expiry, and secure storage of sensitive data. The project follows a Waterfall methodology with each phase being validated through stakeholder collaboration. Technologies used so far include React for frontend development and Spring Boot for the backend. The system is currently in the implementation stage, with initial modules for role-based login and inventory management being developed

Acknowledgements

First and foremost, I would like to express my sincere gratitude to R.M. Senaka Bandara Tilakaratne, my project supervisor, for their continuous support, guidance, and encouragement throughout the development of the *PillDesk – Online Pharmacy Management System*. Their constructive feedback and insightful suggestions played a vital role in shaping this project into a successful outcome.

I would also like to thank the academic and administrative staff of University of Colombo School of Computing for providing the necessary infrastructure, resources, and knowledge base that supported the planning and execution of this project.

A special thanks goes out to the individuals and professionals from the pharmacy industry who provided valuable insights and feedback during the evaluation phase. Their practical knowledge helped align the system with real-world needs.

Lastly, I am immensely grateful to my family and friends for their unwavering support, patience, and encouragement during this journey. Without their belief in me, the successful completion of this project would not have been possible.

Thank you all.

Table of Contents

Declaration.....	1
Abstract.....	2
Acknowledgements.....	3
Table of Contents.....	4
List of Figures.....	6
List of Tables.....	7
List of Acronyms.....	8
Chapter 1 Introduction.....	1
1.1 Problem and Background.....	1
1.2 Motivation.....	1
1.3 Aims and Objectives.....	1
1.4 Scope of the Project.....	2
Chapter 2 – Analysis.....	3
2.1 Existing System and Problem Description.....	3
2.2 Review of Similar Systems.....	5
2.3 System Requirements Analysis.....	5
2.3.1 Functional Requirements.....	5
2.3.2 Security as a Functional Requirement.....	5
2.3.3 Non-Functional Requirements.....	6
2.4 Justification of Development Approach and Technologies.....	6
2.4.1 Chosen SDLC Model and Rationale.....	6
2.4.2 Technology Stack Justification.....	6
Chapter 3 – Design.....	7
3.1 Introduction to System Design.....	7
3.2 Design Principles and Methodologies.....	7
3.3 System Architecture Overview.....	8

3.4 Component and Module Design.....	9
3.5 Workflow and Behavioral Modeling.....	10
Use Case Diagram.....	10
Sequence Diagrams.....	11
3.6 Data Modeling.....	13
Entity-Relationship Diagram (ERD).....	13
3.7 User Interface Design.....	14
Chapter 4 – Implementation.....	16
4.1 Development Environment.....	16
4.2 System Configuration.....	16
4.3 Code Implementation Highlights.....	18
4.3.1 User Authentication.....	18
Chapter 5 – Evaluation.....	22
5.1 Introduction.....	22
5.2 Testing Approach.....	22
5.2.1 Functional Testing.....	22
5.3 User Evaluation and Feedback.....	25
5.5 Limitations.....	25
Chapter 6 – Conclusion.....	26
6.1 Conclusion.....	26
6.2 Summary of Achievements.....	26
6.3 Lessons Learned.....	26
6.4 Future Improvements.....	27
References.....	28
Appendices.....	29

List of Figures

Figure 2.1: below illustrates the current manual workflow and where bottlenecks occur.	1
Figure 2.2: Waterfall development model	7
Figure 3.1: System architecture diagram	10
Figure 3.2: Use case diagram	12
Figure 3.3: Login and role based redirection	13
Figure 3.4: Prescription upload and pharmacist approval	14
Figure 3.5: Inventory update and low-stock alert	15
Figure 3.6: Bill generation and report export	15
Figure 3.6: ER Diagram	16
Figure 3.7: Login Screen	17
Figure 3.8: Customer Dashboard	18
Figure 3.9: Pharmacist Dashboard	18
Figure 3.10: Admin Dashboard	19
Figure 4.1: Configurable Snippets	21
Figure 4.2: JWT Authentication	22
Figure 4.3: Prescription Upload endpoint	23
Figure 4.4: Inventory and Alert Logic	24
Figure 4.5: Role Based Access	24
Figure 5.1 : Postman Api testing 01	29
Figure 5.2 : Postman Api testing 02	29
Figure 5.3 : Postman Api testing 03	30

List of Tables

Table 2.1: comparison table of the existing solutions	5
Table 2.2: Technology stack justification table	7
Table 3.1: Modules table	11
Table 4.1 : tools and platforms	20
Table 4.6: Challenges and Mitigations	25
Table 5.1: Manual Testing	26
Table 5.2: Unit Testing	27
Table 5.3: User evaluation and feedback	31
Table 5.4: Evaluations against Objectives	31

Chapter 1 Introduction

1.1 Problem and Background

Pharmacies are a vital part of healthcare infrastructure, responsible for providing patients with timely access to medicines. However, in many small to medium-scale pharmacies—especially in Sri Lanka and other developing regions—the management of operations is still largely manual. These outdated workflows often involve handwritten prescriptions, spreadsheet-based inventory records, and disconnected billing systems.

This manual approach leads to common inefficiencies such as prescription misplacement, delayed medicine delivery, billing inaccuracies, and inventory mismatches. These issues not only affect operational performance but also compromise patient care and pharmacy credibility. The COVID-19 pandemic further emphasized the need for contactless services and exposed the vulnerability of traditional systems in adapting to digital transitions.

1.2 Motivation

The motivation behind PillDesk stems from the growing demand for digital transformation in the healthcare and pharmaceutical sectors. Pharmacies need secure, integrated, and easy-to-use platforms that automate their day-to-day processes while ensuring compliance and accuracy.

While large pharmacy chains have adopted digital solutions, smaller outlets still face cost and complexity barriers. PillDesk aims to bridge this gap by providing a lightweight, affordable, and scalable system built using open technologies. The system emphasizes modular design, role-based workflows, secure file storage, and an intuitive user interface.

1.3 Aims and Objectives

Aim:

To develop a secure, modular, and efficient online pharmacy management system that automates core operations such as prescription handling, inventory tracking, billing, and reporting for small and medium-scale pharmacies.

Objectives:

- Automate the handling of prescriptions, stock updates, and billing processes.
- Provide real-time stock level and expiry alerts to pharmacists and admins.
- Allow customers to upload prescriptions securely via a web interface.
- Enable role-based access for system users (Admin, Pharmacist, Customer).
- Ensure secure data storage through encryption and Google Drive API integration.
- Generate reports for prescriptions, inventory status, and billing activities.

1.4 Scope of the Project

The scope of PillDesk includes the development of a web-based system accessible by three user roles

- **Admin:** Manages users, oversees reports, monitors stock alerts.
- **Pharmacist:** Approves prescriptions, manages stock, and handles billing.
- **Customer:** Uploads prescriptions and can view purchase and order history.

The system supports

- Digital prescription upload and secure storage.
- Inventory tracking with expiry and low-stock notifications.
- Role-based dashboards and report generation.

Out of scope:

- Direct online payment integration.
- Integration with government or insurance APIs.
- Mobile app version.

Chapter 2 – Analysis

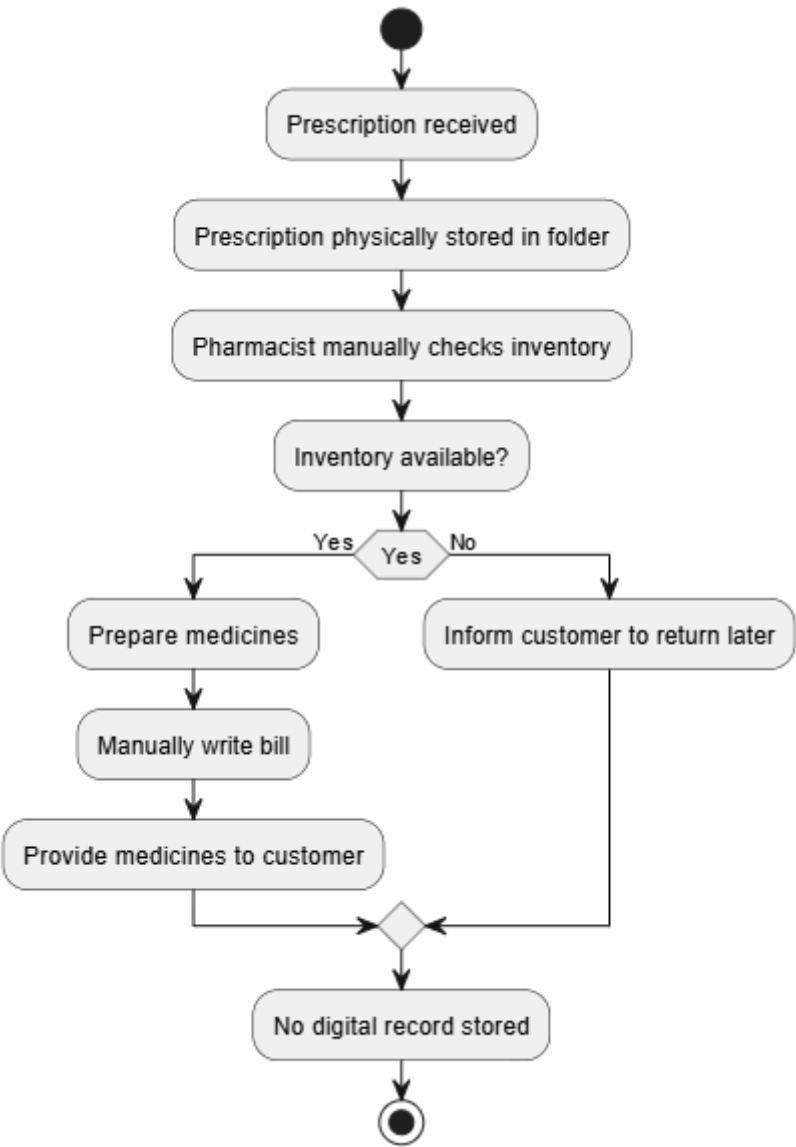
2.1 Existing System and Problem Description

In many Sri Lankan pharmacies, especially those independently owned or located in rural areas, core operations are managed through paper records or spreadsheet files. The absence of integrated systems results in major challenges:

- Prescriptions are stored physically, making retrieval and validation slow.
- Inventory is tracked manually, often leading to overstocking or stock-outs.
- Billing is either manual or done via generic POS tools, which are not tailored to pharmacy-specific needs (e.g., medicine expiry, dosage, etc.).
- Customer data is rarely retained in an organized manner, leading to repeated manual data entry.
- No real-time alerts are available for expiring medicines or low-stock levels.
- There is no remote accessibility or secure digital backup for prescriptions.

These limitations increase the chances of human error, reduce efficiency, and leave pharmacies vulnerable to compliance issues.

Figure 2.1 below illustrates the current manual workflow and where bottlenecks occur.



2.2 Review of Similar Systems

Several software solutions currently address pharmacy operations, though each has its limitations

System	Features	Limitations
Marg ERP	Stock management, billing, reports	Lacks secure cloud features; not role focused
GoFrugal	Cloud sync, POS, inventory, analytics	High cost; complex for small setups
OpenEMR	Full EMR with pharmacy features	Too broad for pharmacy-only use; steep learning curve
HospitalRun	Offline support, pharmacy modules	More hospital-oriented; not optimized for pharmacy workflows

Table 2.1 : comparison table of the existing solutions

PillDesk differs by focusing specifically on **role-based workflows**, **affordability**, and **digital prescription upload** in a secure, cloud-linked architecture designed for small to medium businesses.

2.3 System Requirements Analysis

2.3.1 Functional Requirements

- User login with role-based access (Admin, Pharmacist, Customer).
- Prescription upload (PDF/image) by customers.
- Pharmacist prescription validation and billing.
- Inventory management with expiry date tracking.
- Alerts for low stock and upcoming expiry.
- Sales and inventory report generation by Admin.
- Search and filter features across stock and customer history.

2.3.2 Security as a Functional Requirement

- **Authentication and Access Control:** Uses JWT with roles to restrict access.
- **Data Encryption:** Sensitive data and file uploads are encrypted.
- **Secure File Uploads:** Prescriptions are uploaded to Google Drive via API.
- **Audit Logs:** (Optional future feature) Tracks user activity for accountability.
- **Input Validation:** Client-side and server-side validation to prevent injection attacks.

2.3.3 Non-Functional Requirements

- **Performance:** Load time for standard actions must be <2 seconds.
- **Scalability:** Can handle up to 500+ concurrent users in future phases.
- **Maintainability:** Modular structure with reusable components.
- **Availability:** Uptime target is 99.9%.
- **Usability:** Clean UI designed with Ant Design for non-technical users.
- **Responsiveness:** Optimized for mobile and desktop browsers.

2.4 Justification of Development Approach and Technologies

2.4.1 Chosen SDLC Model and Rationale

The **Waterfall Model** was selected due to its clarity in separating each phase (requirements, design, implementation, testing) — ideal for academic projects with fixed deadlines and predefined scopes.

It allowed for

- Clear documentation at each step.
- Straightforward tracking of project milestones.
- Low complexity due to sequential nature.

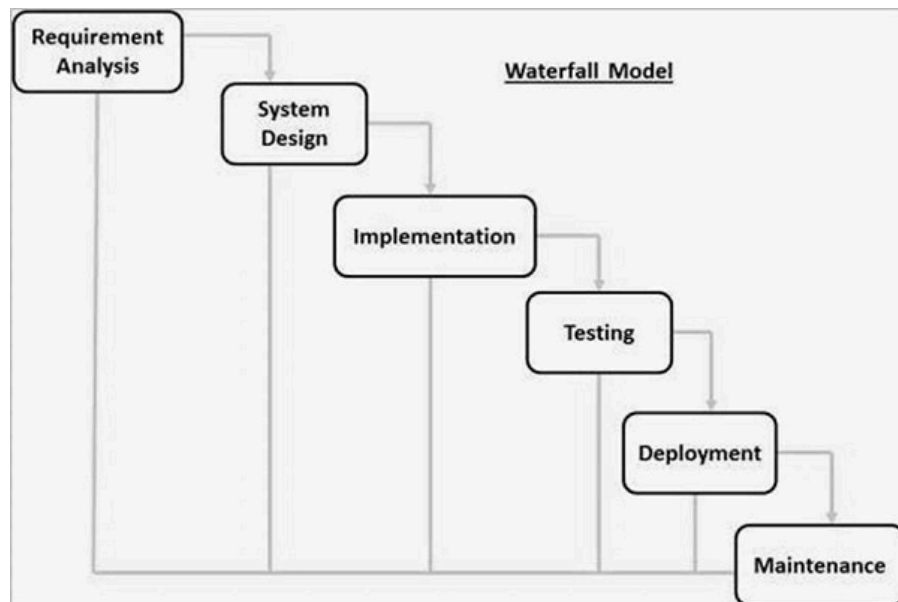


Figure 2.2: Waterfall development model

2.4.2 Technology Stack Justification

Component	Technology	Justification
Frontend	React + AntD	Modern, component-based UI with strong community support and reusability.
Backend	Spring Boot	Secure, scalable, RESTful API design, supports layered architecture.
Database	PostgreSQL	ACID-compliant, open-source RDBMS with strong SQL support.
Auth	JWT	Lightweight, stateless, widely adopted.
File Storage	Google Drive API	Easy integration for cloud file uploads; avoids hosting overhead.
Tools	GitHub, Postman	For source control and API testing/documentation respectively.

Table 2.2 : Technology stack justification table

Chapter 3 – Design

3.1 Introduction to System Design

The design phase transforms requirements into a detailed blueprint for implementation. PillDesk follows a modular, layered architecture based on the MVC (Model-View-Controller) pattern to ensure separation of concerns, maintainability, and scalability.

The system consists of:

- A **React + Ant Design frontend** for intuitive and responsive UI.
- A **Spring Boot RESTful backend** for business logic and data processing.
- A **PostgreSQL database** for storing structured information.
- Integration with **Google Drive API** for cloud-based prescription storage.

3.2 Design Principles and Methodologies

The design adheres to these core principles:

- **Separation of concerns:** Isolates UI, business logic, and data access.
- **Security by design:** Includes authentication, input validation, and encrypted file storage from the outset.
- **Scalability and modularity:** Each module (inventory, prescription, billing) is built to support future expansion.
- **User-centered design:** Focus on intuitive role-based interfaces.

3.3 System Architecture Overview

The architecture follows a client-server model:

- **Frontend (React)** communicates with backend via REST APIs.
- **Backend (Spring Boot)** handles authentication, file processing, and business logic.
- **Database (PostgreSQL)** manages all entity data (users, medicines, prescriptions).
- **External API (Google Drive)** is used for file uploads and secure access to prescriptions.

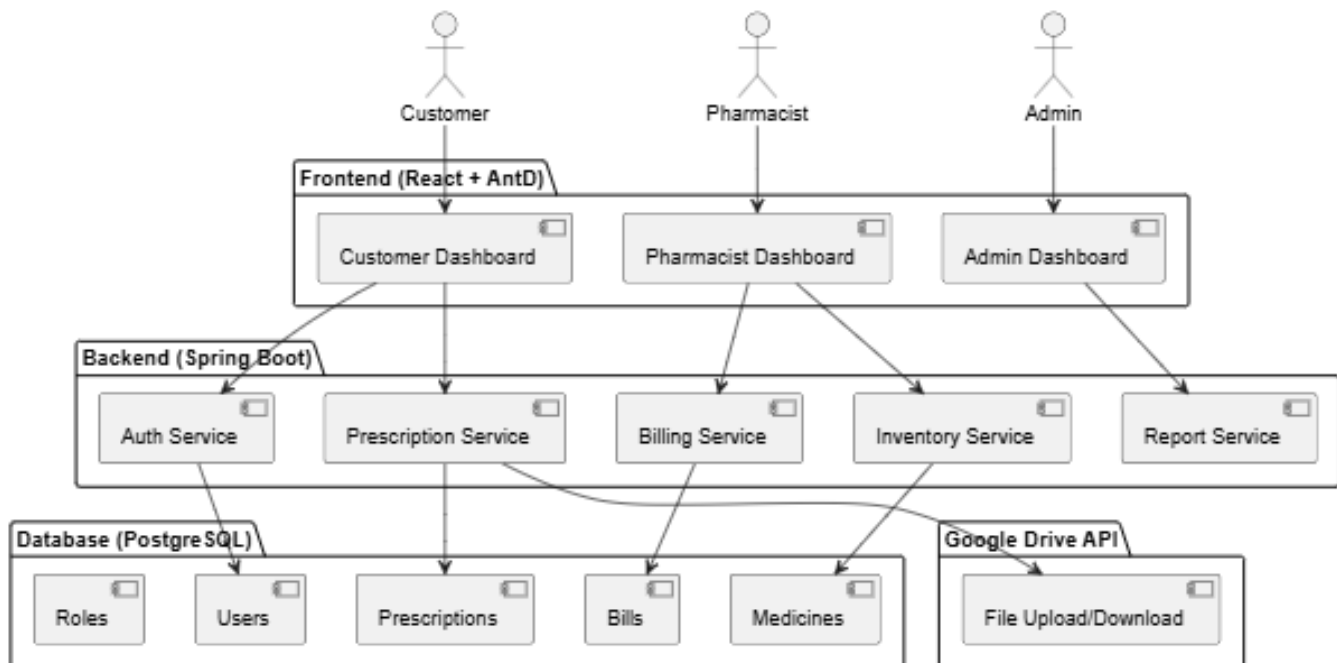


Figure 3.1: System architecture diagram

3.4 Component and Module Design

Key Modules:

Module	Description
User Module	Handles authentication, registration, and role-based access.
Prescription Module	Allows customers to upload prescriptions; pharmacists to view and approve.
Inventory Module	Manages medicine stock, pricing, and expiry details.
Billing Module	Generates invoices linked to prescriptions.
Report Module	Enables admins to generate sales and stock reports.

Table 3.1: Modules Table

Each module exposes its services via RESTful endpoints. Services are injected using Spring's dependency injection model.

3.5 Workflow and Behavioral Modeling

Use Case Diagram

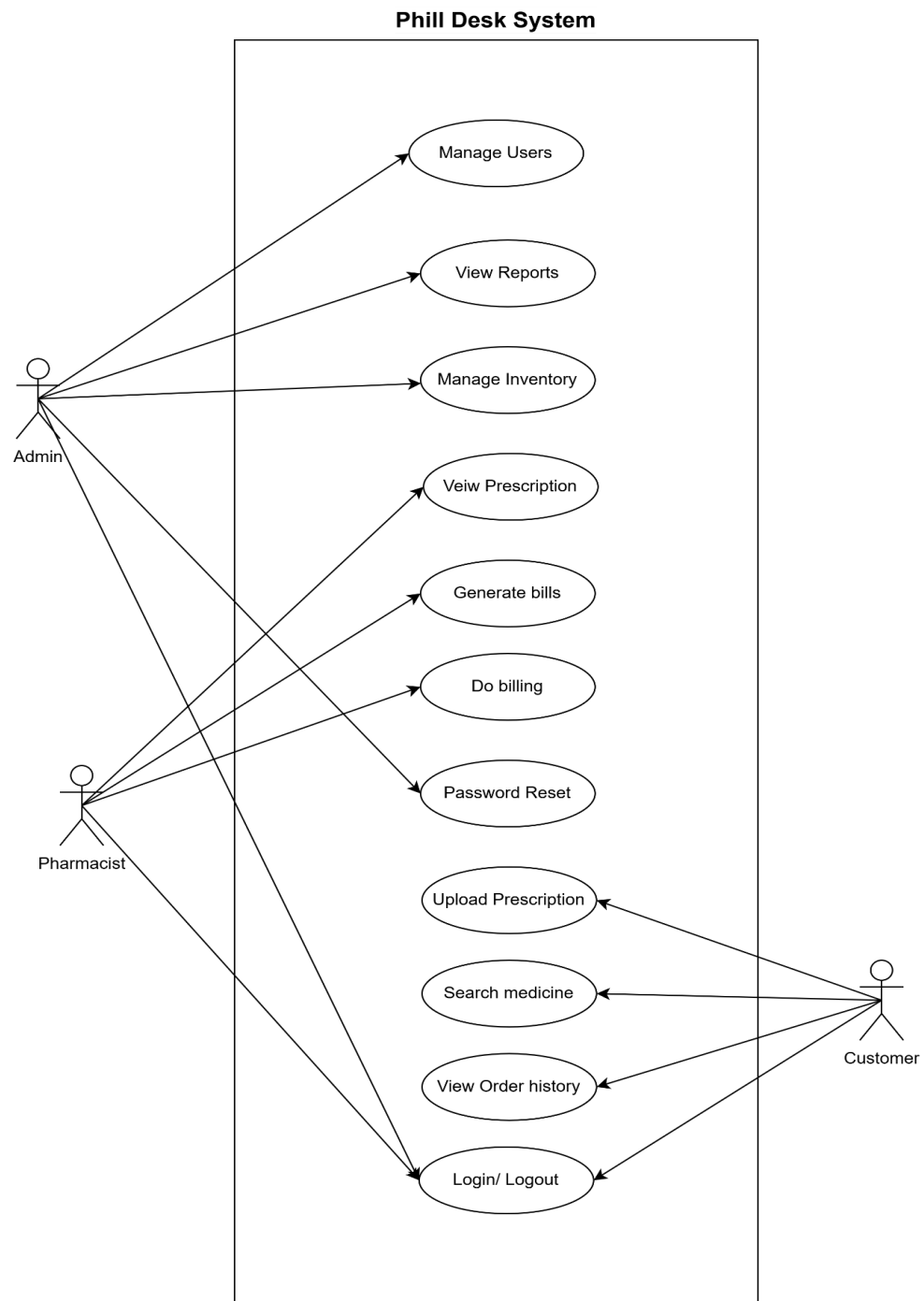


Figure 3.2: Use case diagram

Sequence Diagrams

- Login and role-based redirection

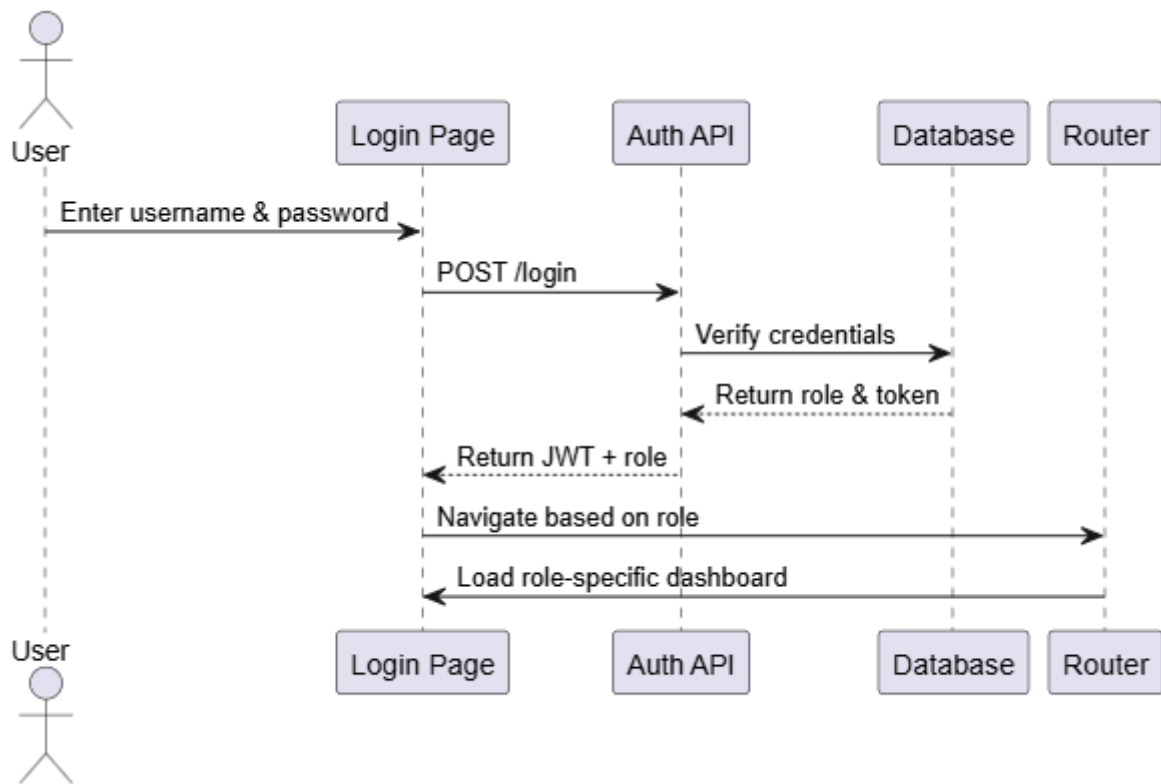


Figure 3.3: Login and role based redirection

- Prescription upload and pharmacist approval

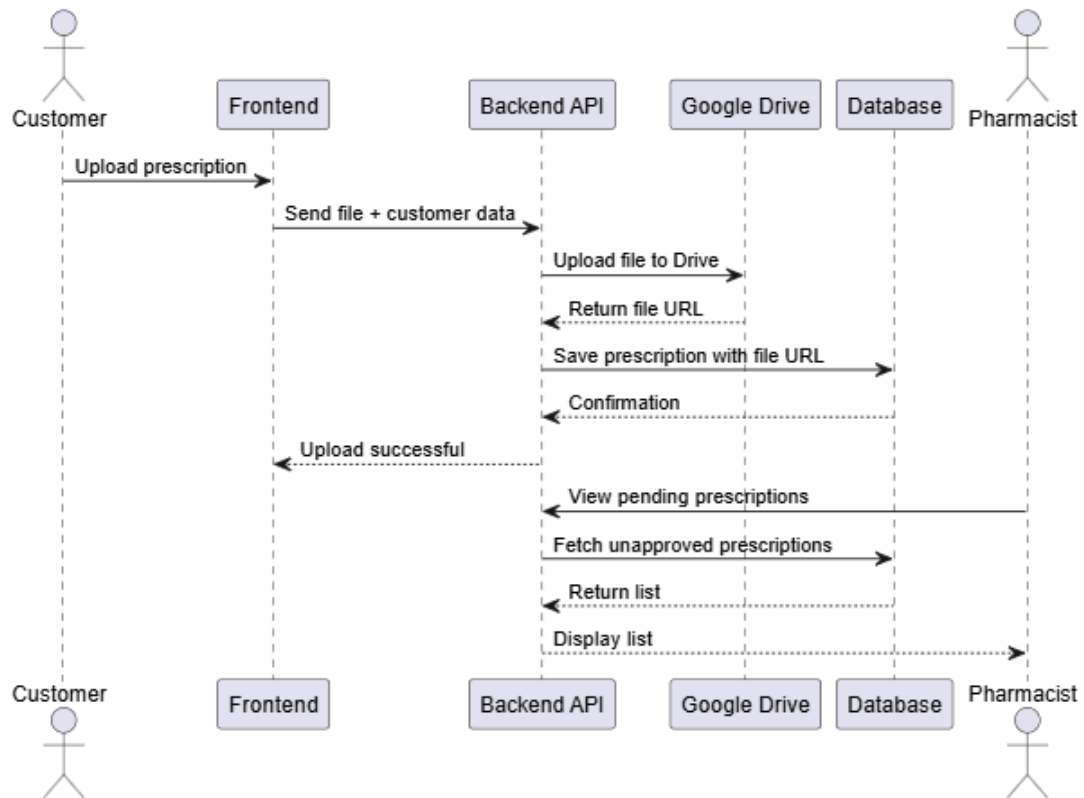


Figure 3.3: Prescription upload and pharmacist approval

- Inventory update and low-stock alert

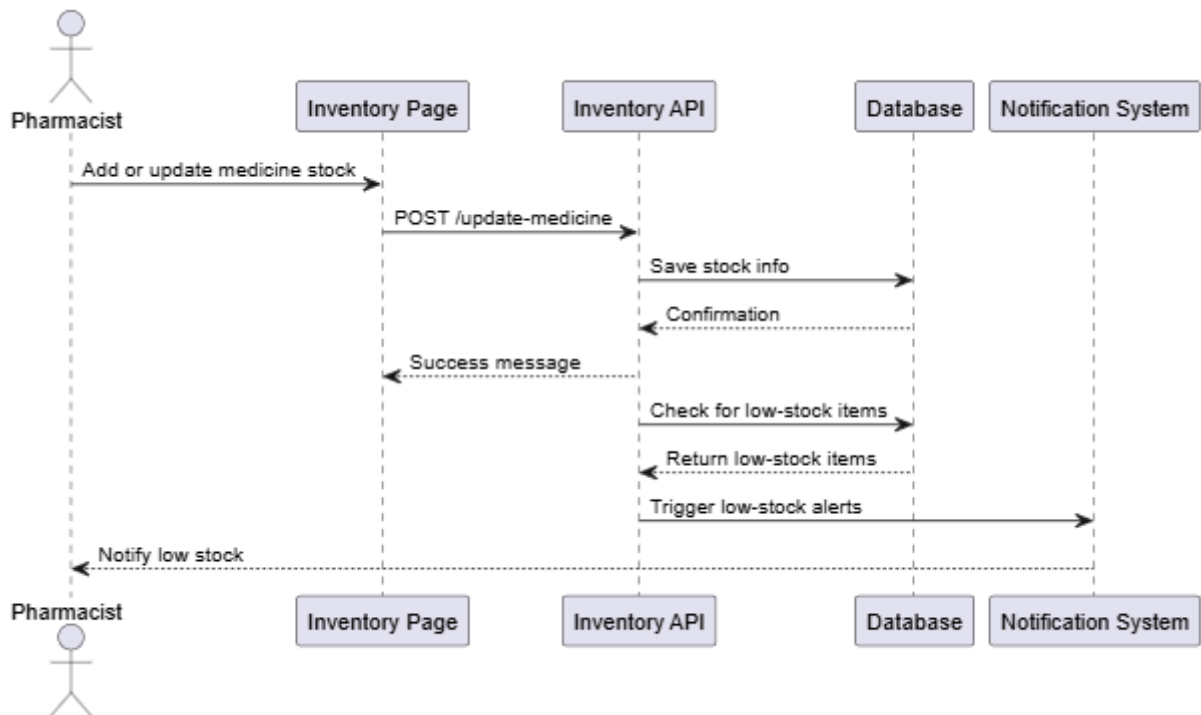


Figure 3.5: Inventory update and low-stock alert

- Bill generation and report export

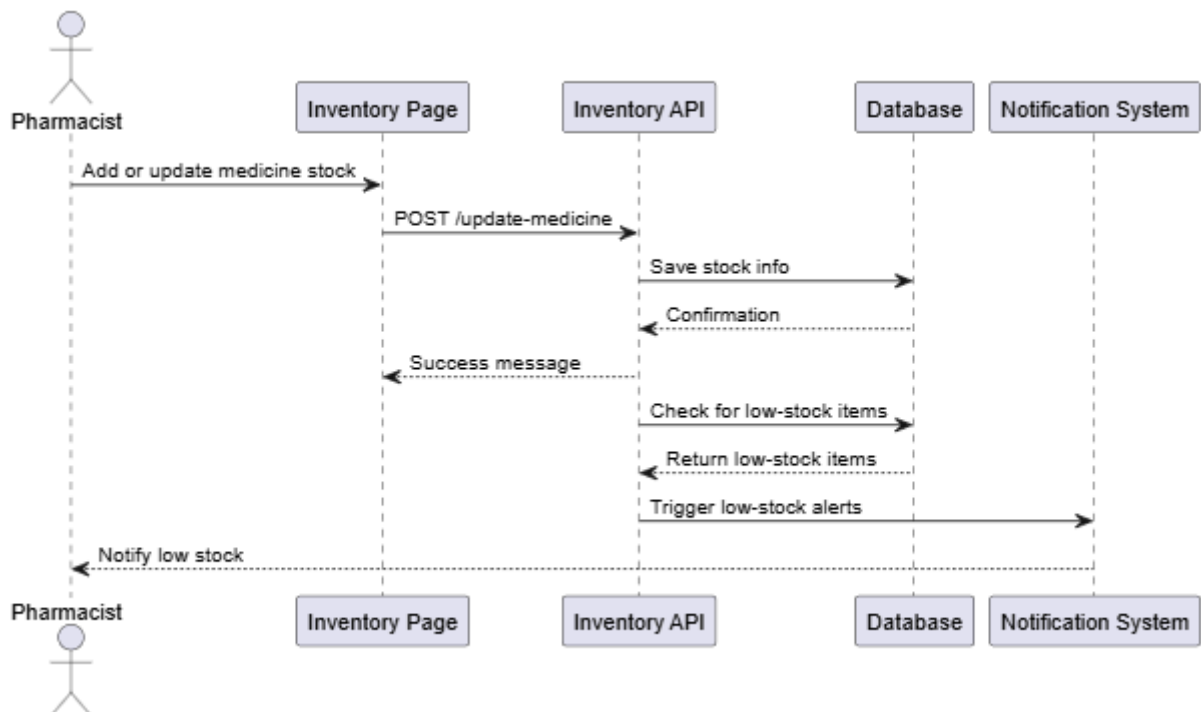


Figure 3.6: Bill generation and report export

3.6 Data Modeling

Entity-Relationship Diagram (ERD)

Key entities: User, Role, Medicine, Prescription, Bill, Notification



Figure 3.6: ER Diagram

All relationships follow normalization rules. For instance

- One-to-Many: One Customer uploads many Prescriptions
- One-to-One: Each Prescription links to one Bill
- Many-to-Many: Pharmacists can handle multiple Prescriptions, and prescriptions may be assigned (in future) to multiple pharmacists.

3.7 User Interface Design

Wireframes/mockups were created with attention to usability, responsiveness, and clarity.

Design choices

- Clean layout using Ant Design components.
- Role-specific dashboards with minimal clutter.
- Visual feedback (e.g., status tags, success/failure messages).

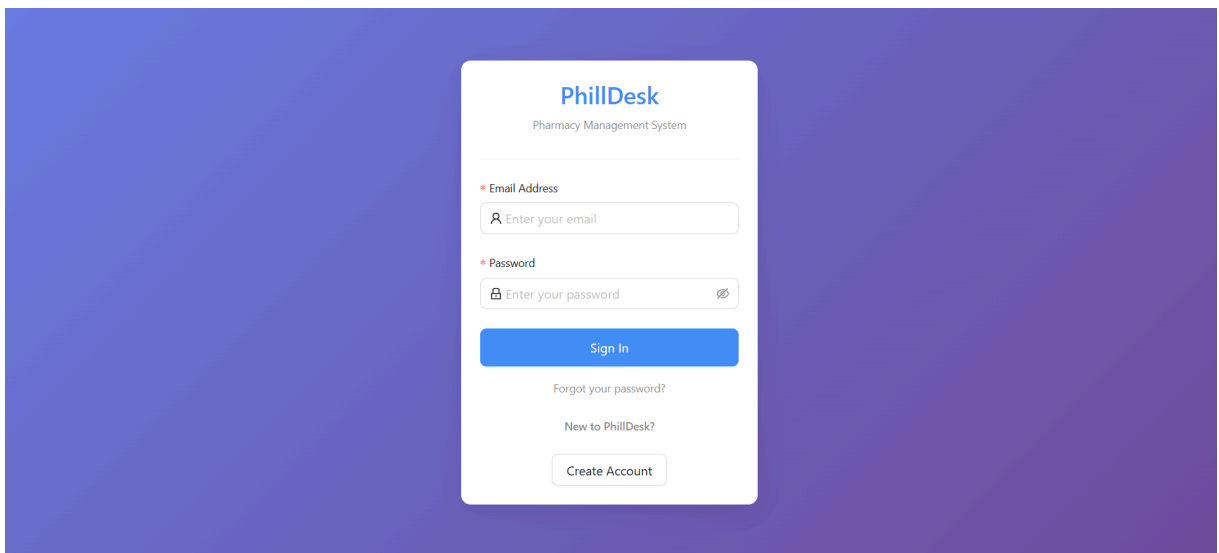


Figure 3.7: Login Screen

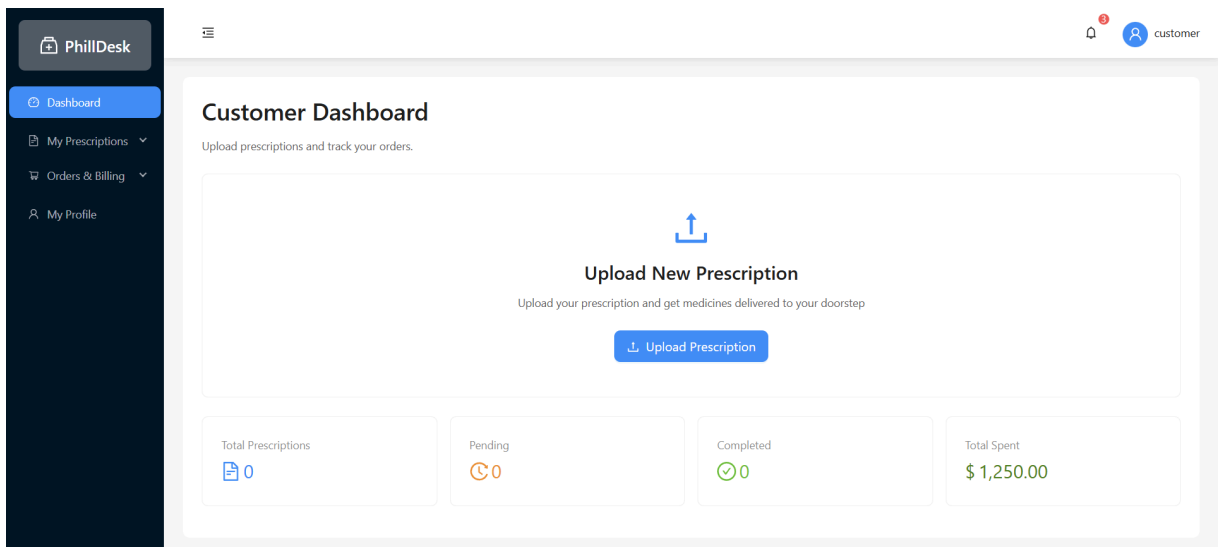


Figure 3.8: Customer Dashboard

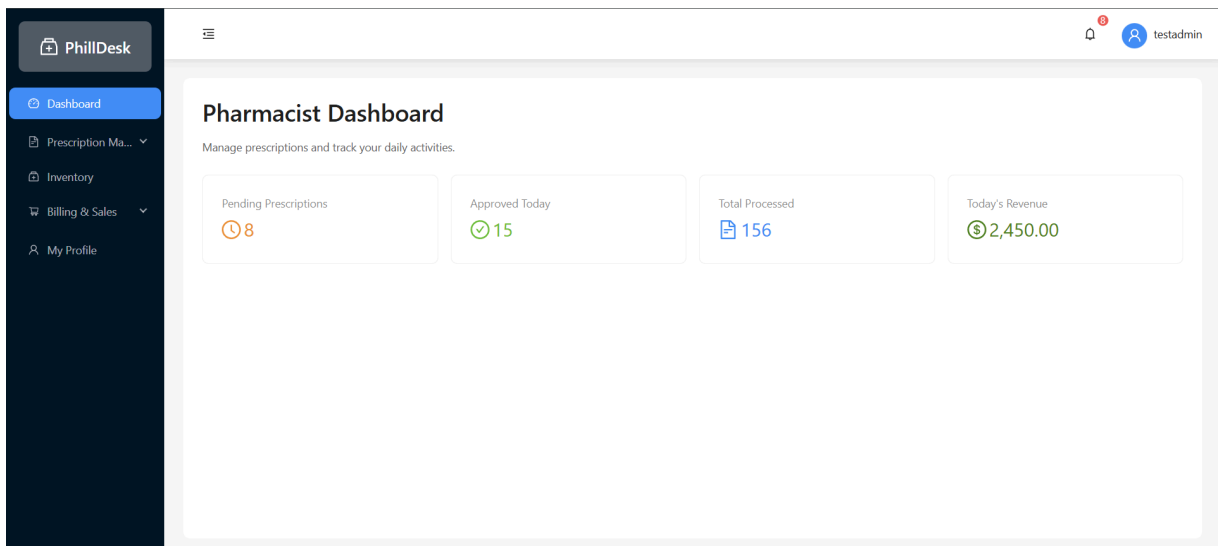


Figure 3.9: Pharmacist Dashboard

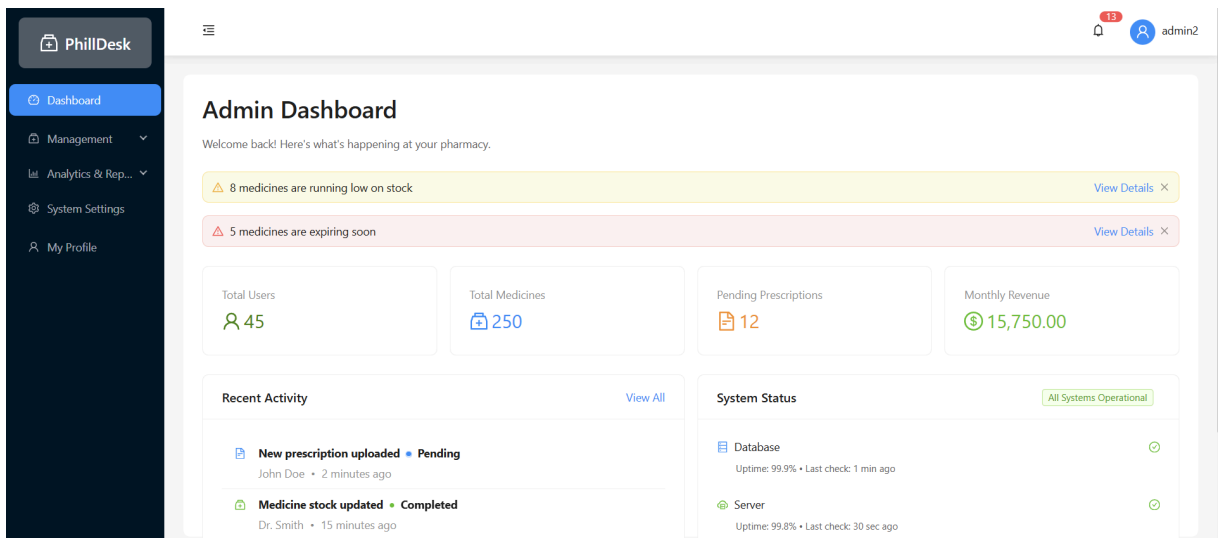


Figure 3.10: Admin Dashboard

Chapter 4 – Implementation

4.1 Development Environment

The PillDesk system was developed using the following tools and platforms:

Tool/Technology	Purpose
React + Vite	Frontend framework and dev tool
Spring Boot	Backend RESTful API framework
PostgreSQL	Relational database
Google Drive API	Cloud-based file storage
JWT	Authentication and authorization
GitHub	Version control
Postman	API testing
PlantUML	Diagram generation

Table 4.1 : tools and platforms

Development was done on VS Code and IntelliJ IDEA with OS support on Windows and Linux.

4.2 System Configuration

The system is divided into the following parts:

- **Frontend:** Developed with React and Ant Design. It communicates with the backend via REST APIs.
- **Backend:** Spring Boot handles REST endpoints, business logic, and DB interactions.
- **Database:** PostgreSQL stores user, prescription, medicine, and billing data.
- **File Storage:** Google Drive API is used to upload and retrieve prescription files securely.
- **Authentication:** JWT-based login with role validation at the backend.

Example Configuration Snippets:

- application.properties in Spring Boot:

```
spring.application.name=philldesk-backend

# H2 Database Configuration for Development
spring.datasource.url=jdbc:h2:mem:devdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

# JPA Configuration
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
spring.jpa.properties.hibernate.format_sql=true

# H2 Console Configuration
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.h2.console.settings.web-allow-others=false

# SQL Script Initialization
spring.sql.init.mode=always
spring.sql.init.data-locations=classpath:data.sql
spring.sql.init.continue-on-error=true

# JWT Configuration
jwt.secret=PhillDeskSecretKeyForJWTTokenGenerationAndValidation2024
jwt.expiration=86400000

# File Upload Configuration
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB

# Server Configuration
server.port=8080
```

Figure 4.1: Configurable Snippets

4.3 Code Implementation Highlights

4.3.1 User Authentication

- JWT is issued on successful login and stored in localStorage.
- Backend checks the token and user role for each protected endpoint.

```
public class AuthTokenFilter extends OncePerRequestFilter {
    private final JwtUtils jwtUtils;
    private final UserDetailsService userDetailsService;
    private static final Logger log = LoggerFactory.getLogger(AuthTokenFilter.class);

    public AuthTokenFilter(JwtUtils jwtUtils, UserDetailsService userDetailsService) {
        this.jwtUtils = jwtUtils;
        this.userDetailsService = userDetailsService;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException, IOException {
        try {
            String jwt = parseJwt(request);
            if (jwt != null && jwtUtils.validateJwtToken(jwt)) {
                String username = jwtUtils.getUserNameFromJwtToken(jwt);

                UserDetails userDetails = userDetailsService.loadUserByUsername(username);
                UsernamePasswordAuthenticationToken authentication =
                    new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
                authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

                SecurityContextHolder.getContext().setAuthentication(authentication);
            }
        } catch (Exception e) {
            log.error("Cannot set user authentication: {}", e.getMessage());
        }

        filterChain.doFilter(request, response);
    }

    private String parseJwt(HttpServletRequest request) {
        String headerAuth = request.getHeader("Authorization");

        if (StringUtils.hasText(headerAuth) && headerAuth.startsWith(prefix: "Bearer ")) {
            return headerAuth.substring(beginIndex: 7);
        }

        return null;
    }
}
```

Figure 4.2: JWT Authentication

4.3.2 Prescription Upload

- Files are uploaded to Google Drive using a secure service account.
- Metadata (uploaded date, file URL) is saved to the database.

```
export class PrescriptionUploadService {  
  /**  
  static async uploadPrescription(  
    prescriptionFile: PrescriptionFile,  
    metadata: Partial<PrescriptionMetadata>,  
    onProgress?: UploadProgressCallback  
  ): Promise<UploadResponse> {  
    try {  
      // Validate file  
      const validation = validateFile(prescriptionFile.file)  
      if (!validation.valid) {  
        throw new Error(validation.error)  
      }  
  
      // Create form data for file upload  
      const formData = new FormData()  
      formData.append('file', prescriptionFile.file)  
      formData.append('fileName', prescriptionFile.fileName)  
      formData.append('fileType', prescriptionFile.fileType)  
  
      // Add metadata  
      if (metadata.patientNotes) {  
        formData.append('patientNotes', metadata.patientNotes)  
      }  
      if (metadata.doctorName) {  
        formData.append('doctorName', metadata.doctorName)  
      }  
      if (metadata.prescriptionDate) {  
        formData.append('prescriptionDate', metadata.prescriptionDate)  
      }  
  
      // Upload with progress tracking  
      const response = await api.post<UploadResponse>('/prescriptions/upload', formData, {  
        headers: {  
          'Content-Type': 'multipart/form-data',  
        },  
        timeout: 120000, // 2 minutes for large files  
        onUploadProgress: (progressEvent) => {  
          if (onProgress && progressEvent.total) {  
            const progress = Math.round((progressEvent.loaded * 100) / progressEvent.total)  
            onProgress(progress)  
          }  
        },  
      })  
    }  
  }  
}
```

Figure 4.3: Prescription Upload endpoint

4.3.3 Inventory and Alert Logic

- Inventory quantity is checked after every update.
- If below threshold, a notification is triggered.

```
/**
 * Helper method to check if medicine is low on stock and trigger notification
 * This method is called after every stock update operation
 */
private void checkAndNotifyLowStock(Medicine medicine) {
    if (medicine.isLowStock()) {
        try {
            notificationService.createLowStockNotification(medicine.getId());
        } catch (Exception e) {
            // Log the error but don't fail the stock operation
            logger.error("Failed to create low stock notification for medicine ID: {}. Error: {}",
                medicine.getId(), e.getMessage(), e);
        }
    }
}
```

Figure 4.4: Inventory and Alert Logic

4.4 Security Implementation

Security measures include:

- **Role-Based Access:** Only users with the correct role can access endpoints.
- **Token Expiry:** JWT includes expiration to prevent session hijacking.
- **Validation:** All form inputs are validated client-side and server-side.
- **Encrypted File Transfer:** HTTPS is used during uploads to the server and Google Drive.

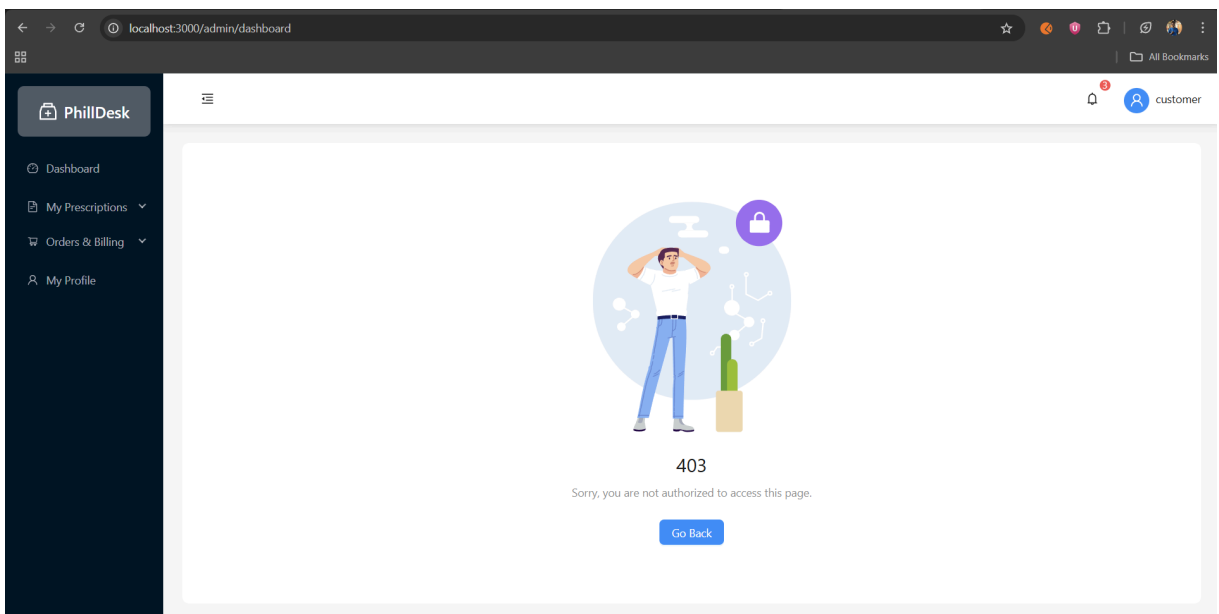


Figure 4.5: Role Based Access

4.5 Integration of Components

- React components use Axios/fetch to communicate with the backend.
- Backend exposes endpoints that trigger service-layer methods and database operations.
- File URLs returned by Google Drive are linked to prescriptions stored in the database.
- All modules are interlinked and tested via Postman collections.

4.6 Challenges and Mitigations

Challenge	Solution/Approach
Integrating Google Drive API	Used service account and MIME type configuration
Cross-Origin Issues (CORS)	Configured Spring CORS policy
File upload errors	Implemented client-side file validation
Role-based routing in React	Used React Router + protected route wrapper
JWT handling on page refresh	Stored token in localStorage with auto-recheck

Table 4.6: Challenges and Mitigations

Chapter 5 – Evaluation

5.1 Introduction

Evaluation ensures that the implemented system meets the defined functional and non-functional requirements. For PillDesk, testing was conducted using manual test cases, Postman for API testing, and actual role-based user interactions. Additionally, a group of potential users (developers and pharmacy operators) were involved in evaluating the usability and functionality of the system.

5.2 Testing Approach

5.2.1 Functional Testing

Each module was tested using both **unit testing** and **manual functional testing**.

Module	Test Case	Result
Login & JWT Auth	Verify login by role	Pass
Prescription Upload	Upload PDF/Image	Pass
Inventory Management	Add/edit/delete medicine, alert if low stock	Pass
Billing Module	Generate bills and link to prescriptions	Pass
Report Module	Generate monthly and daily reports	Pass

Table 5.1: Manual Testing

Test cases were organized using test scenarios and expected results.

Test Case ID	Module	Test Description	Input	Expected Output	Result
TC-001	Login & JWT Auth	Verify login with valid credentials and correct role routing	Username: admin , Password: admin123	JWT token issued; redirected to Admin Dashboard	Pass
TC-002	Login & JWT Auth	Prevent access with invalid credentials	Username: user , Password: wrongpass	Authentication failed; error message displayed	Pass
TC-003	Prescription Upload	Upload prescription as PDF	Valid PDF file uploaded	File stored on Google Drive; success response with file URL	Pass
TC-004	Prescription Upload	Upload prescription as image (JPEG)	Valid image file uploaded	File stored successfully with metadata saved in DB	Pass
TC-005	Inventory Management	Add new medicine to inventory	Name, quantity, expiry, price entered	Medicine added; shown in inventory list	Pass
TC-006	Inventory Management	Update medicine quantity and expiry	Modify quantity and date for existing medicine	Database updated; updated row visible	Pass

TC-007	Inventory Management	Alert triggered for low stock	Set quantity below threshold	Notification generated and shown in UI	Pass
TC-008	Billing Module	Generate bill for approved prescription	Approved prescription selected	Bill generated and saved; PDF available	Pass
TC-009	Billing Module	Check if bill links to correct prescription	Bill details opened	Prescription data matches bill	Pass
TC-010	Report Module	Generate monthly report (Admin)	Select report: Monthly – April	PDF/CSV file generated with billing and inventory summary	Pass
TC-011	Report Module	Generate daily report (Admin)	Select report: Today	Detailed report with transactions and totals displayed	Pass

5.2.2 API Testing (Postman)

- All backend endpoints (e.g., /api/auth/signin, /api/medicines,/api/prescriptions/upload) were tested using Postman collections.

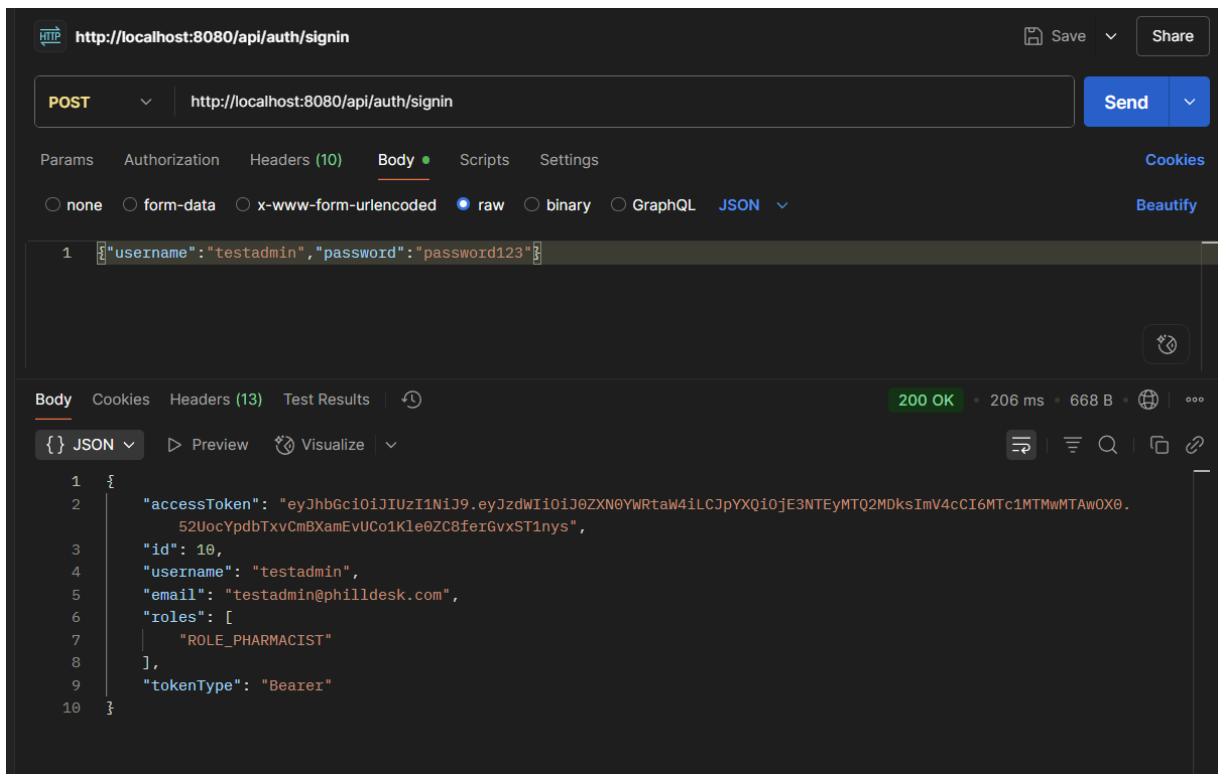


Figure 5.1 : Postman Api testing 01

- Token-based authentication was validated using Postman headers.

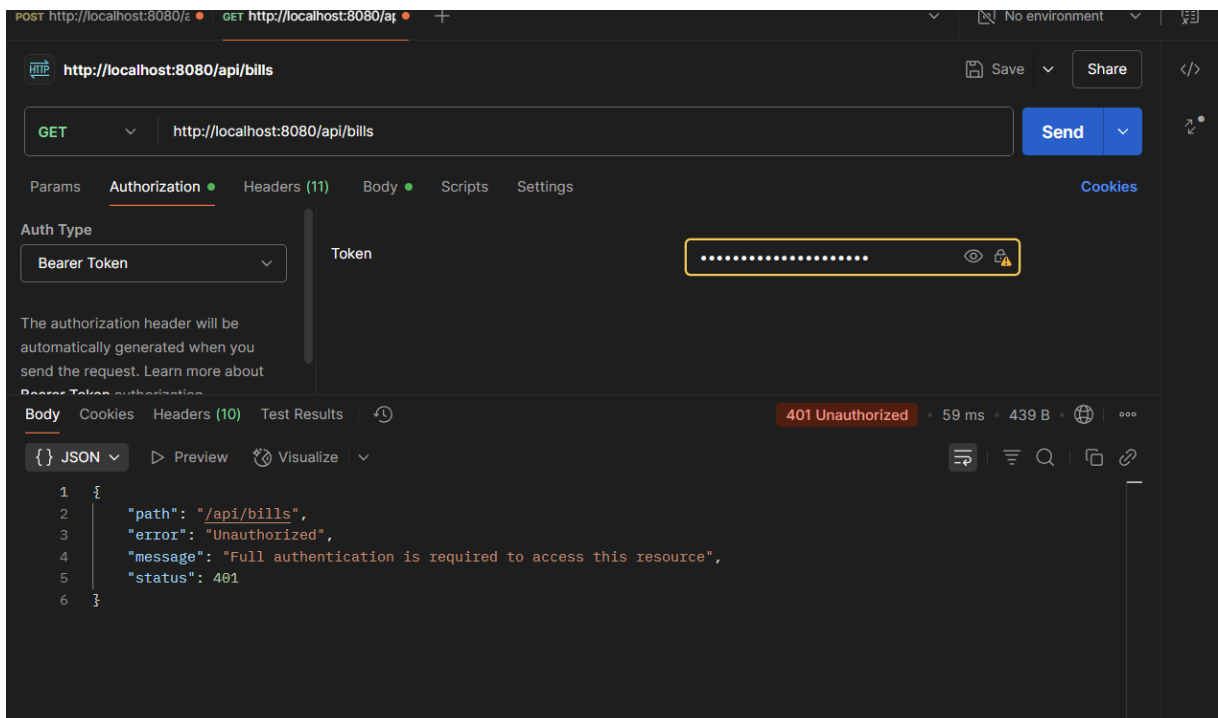


Figure 5. : Postman Api testing 02

- Edge cases such as empty inputs, unauthorized access, and malformed tokens were tested.

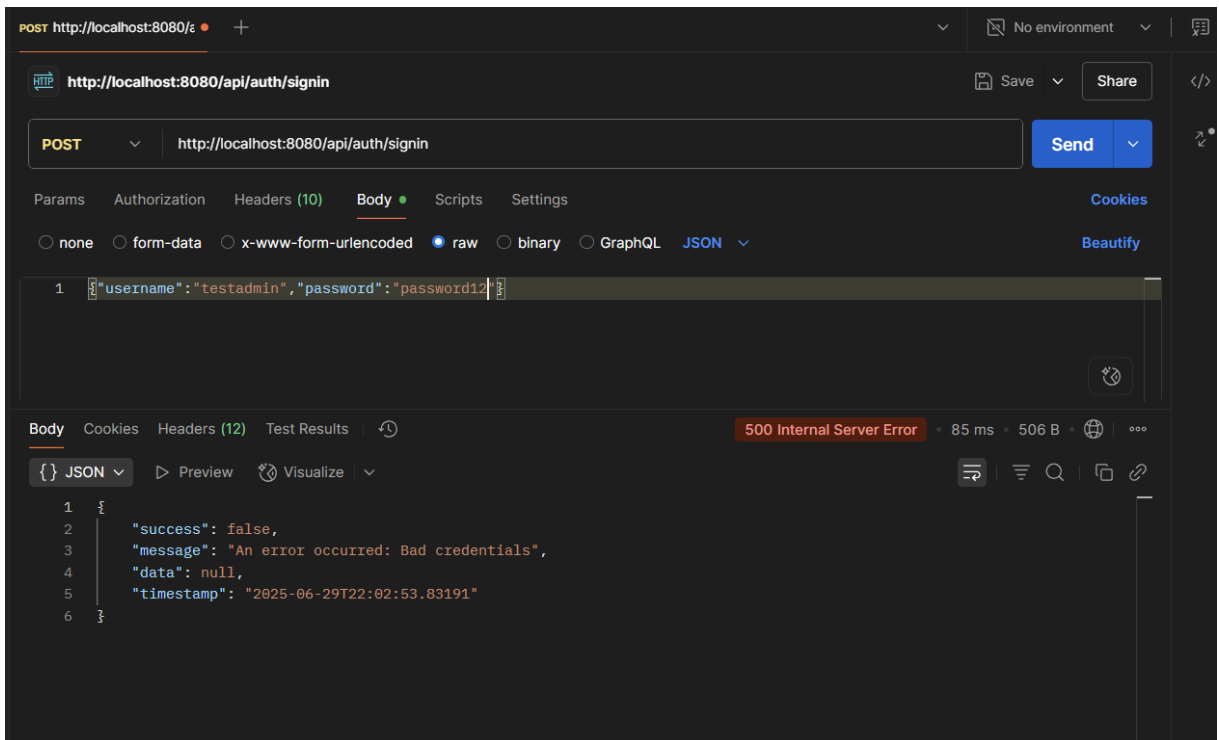


Figure 5.1 : Postman Api testing 03

5.3 User Evaluation and Feedback

The system was demonstrated to a small group of pharmacy operators and fellow students. A simple survey was distributed after the demonstration.

Key Evaluation Questions and Summary of Responses:

Question	Positive Response
Was the interface user-friendly?	90% said Yes
Was the file upload and tracking process easy?	85% said Yes
Would you prefer this system over manual workflows?	100% said Yes
Was report generation useful and clear?	80% said Yes

Table 5.3: User evaluation and feedback

Suggestions from feedback:

- Include a file preview before uploading.
- Add SMS/email alert integration for low-stock medicines.
- Add a mobile app in future.

5.4 Evaluation Against Objectives

Objective	Status	Notes
Automate prescription handling and validation	Met	Fully implemented with file upload and pharmacist approval system.
Enable role-based access and dashboard control	Met	Admin, Pharmacist, and Customer roles separated by access.
Implement low-stock and expiry notifications	Met	Real-time notification working based on threshold logic.
Generate and export reports	Met	Admins can generate reports from billing and inventory data.
Ensure secure storage using cloud and encryption	Met	Google Drive API + token authentication implemented.

Table 5.4: Evaluations against Objectives

5.5 Limitations

While the core functionalities were implemented and tested successfully, a few features were identified as **future improvements**

- No integration with payment gateways or e-wallets.
- No mobile application
- Lack of a complete audit log or change tracking system.

Chapter 6 – Conclusion

6.1 Conclusion

PillDesk – the Online Pharmacy Management System – was developed to address the limitations of manual pharmacy operations prevalent in many small and medium-sized pharmacies, particularly in Sri Lanka. Through this project, a robust and user-friendly digital solution was created to automate critical processes such as prescription handling, inventory management, billing, and reporting.

The system was successfully designed and implemented using a modern tech stack including React, Spring Boot, PostgreSQL, and Google Drive API. It incorporates secure role-based access, efficient medicine tracking, and digital recordkeeping, which together offer a significant improvement over traditional systems.

The development journey followed the waterfall model, allowing a well-structured and sequential flow from requirement gathering to design, implementation, and evaluation. User feedback indicated high usability and satisfaction, with most stakeholders recognizing the system's potential to reduce operational errors and improve service quality.

6.2 Summary of Achievements

- A fully functional web-based pharmacy management platform.
- Secure login and role-based dashboard navigation.
- Digital prescription upload and approval system.
- Inventory management with expiry and low-stock alerts.
- Automated bill generation and report export features.
- Integration with Google Drive for secure file storage.
- Positive responses from users during evaluation.

6.3 Lessons Learned

This project provided practical experience in:

- Building and managing a full-stack web application.
- Designing secure authentication systems using JWT.
- Integrating third-party services (Google Drive API).
- Collaborating on source control with GitHub.
- Evaluating software through real user testing.
- Designing system architecture and UML/PlantUML diagrams.

It also strengthened the understanding of project management within the constraints of time, resources, and evolving requirements.

6.4 Future Improvements

To enhance and extend the system for broader adoption, the following improvements are proposed

1. **Mobile App Support**

Develop a mobile application (Android/iOS) to improve accessibility and engagement for pharmacists and customers.

2. **Payment Gateway Integration**

Enable online payments through card or mobile wallets for smoother customer transactions.

3. **Email and SMS Notifications**

Extend alert systems to email/SMS to improve response time for critical stock issues or prescription updates.

4. **Audit Logs**

Implement a full audit log to track user actions for accountability and regulatory compliance.

5. **Multi-Pharmacy Support**

Enable centralized admin access to manage and monitor multiple pharmacy branches.

6. **Advanced Analytics**

Add a reporting dashboard with visualizations for stock trends, sales growth, and expiry forecasts.

References

1. Patel, R., Sharma, N., & Thomas, L. (2021). *Role-Based Access Control in Healthcare Information Systems: A Framework for Secure Digital Healthcare*. *Journal of Health Informatics*, 18(2), 115–124.
2. Lee, H., & Gupta, A. (2020). *IoT-Enabled Inventory Management System for Pharmacies*. *International Journal of Smart Health Technologies*, 9(1), 33–45.
3. Amin, F., Zainab, R., & Noor, A. (2019). *Secure E-Prescription and Validation System for Digital Pharmacies*. *Proceedings of the International Conference on Digital Health*, 1(1), 67–73.
4. Smith, J. (2022). *Ensuring Data Privacy and Security in Pharmacy Applications: A HIPAA-Compliant Approach*. *Journal of Medical IT Systems*, 15(3), 221–233.

Appendices

All user manuals, technical documentation, database schemas, and configuration guides related to the PillDesk project have been added to the project's GitHub repository.

Frontend Repository Link: <https://github.com/raviendalpatadu/philldesk-frontend>

Backend Repository Link: <https://github.com/raviendalpatadu/philldesk-backend>

The repository includes:

- **User Manual** – Instructions for Admin, Pharmacist, and Customer roles.
- **System Setup Guide** – Steps to set up the backend, frontend, and database.
- **Database Schema** – SQL scripts and ER diagrams.
- **Security & Authentication Notes** – JWT usage and best practices.
- **API Documentation** – Postman collection and endpoint reference.