TITLE OF THE DISSERTATION

K R K Dalpatadu

June 2025

TITLE OF THE DISSERTATION

K R K Dalpatadu

2730271

Name(s) of the Supervisor(s):

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:…….

Name of Candidate: . . . . . .

Countersigned by:

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

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

Abstract

When writing an abstract for a software implementation project dissertation, you should provide a concise summary that captures the essence of the entire project. Begin by briefly stating the problem or need that motivated the development of the software, followed by a clear description of the objective or purpose of the system. Next, outline the main technologies, tools, or methodologies used in the implementation, keeping the details high-level. Highlight the key features or functionalities developed, and then summarize the results or outcomes, such as how well the system performed or what feedback was received from users or testing. Conclude the abstract with a statement on the significance or potential impact of the project, including any suggestions for future enhancements. The abstract should typically be written after the project is complete and must fit to a single page.

Acknowledgements

Briefly thank those who supported you academically and personally. Start by acknowledging your supervisor for their guidance, followed by any staff or institutions that provided help or resources. If relevant, include thanks to an organization involved in your project. You may also express appreciation to peers, family, or friends for their support. Keep the tone formal, sincere, and concise, typically within one short paragraph.

Table of Contents

When preparing a table of contents, use only two levels, list the main chapters as first-level headings (e.g., Introduction, Design, Implementation) and include important subheadings as second-level entries (e.g., 2.1 System Architecture, 2.2 Tools Used). Align page numbers neatly and ensure the numbering matches the actual document. To maintain consistency and accuracy, it is recommended to use the built-in table of contents and heading styles feature in your word processing application (such as Microsoft Word or Google Docs). This allows you to automatically generate and update the table of contents based on the heading levels used in your document. Ensure proper formatting, align page numbers neatly, and keep the layout clean and easy to navigate.

Table of Contents

List of Figures

All figures in the dissertation should be numbered and named using an appropriate caption. Numbering is done using chapter number and a sequence number (e.g. Figure 3.2 for second figure in Chapter 3). Figures in the appendices are numbered using the Appendix letter (e.g. Figure C.2 for second figure in Appendix C). List of figures consists of figure number, captions and page numbers. List can be generated using features of a word processing package. All figures used in the main chapters must be described in text prior to its use and must be referred to using its figure number. For example, in Section 3.5 of this document Figure 3.1 is referred to in text in the paragraph before the figure.

List of Tables

All tables in the dissertation should be numbered and named using an appropriate caption. Numbering is done using chapter number and a sequence number (e.g. Table 3.2 for second table in Chapter 3). Tables in the appendices are numbered using the Appendix letter (e.g. Table C.2 for second table in Appendix C). List of tables consists of table number, captions and page numbers. List can be generated using features of a word processing package. All tables used in the main chapters must be described in text prior to its use and must be referred to using its table number. Example: In section 5.1 of this document Table 5.2 is referred to in text in the paragraph.

List of Acronyms

Provides the meanings of all abbreviations used in the dissertation in alphabetical order. (Note: Common/obvious abbreviations/terms should not be included in the list.)

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 (though mobile browser compatibility is ensured).

Chapter 2 – Analysis

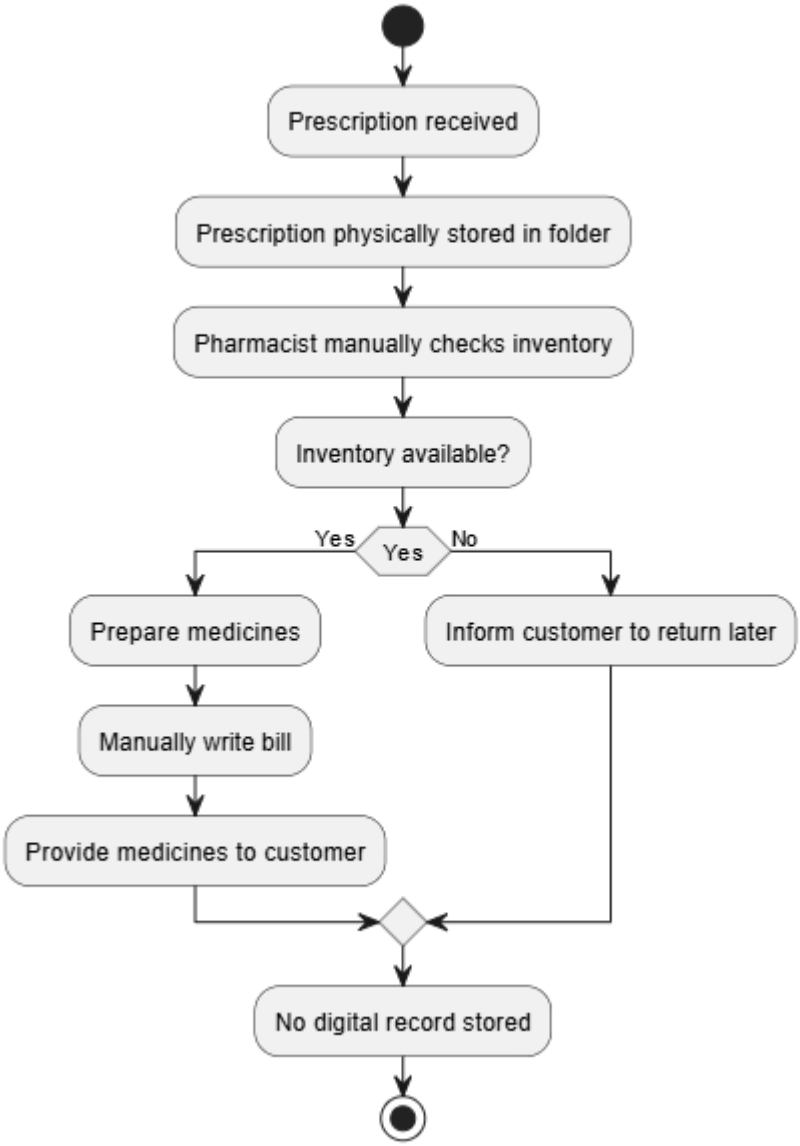
2.1 Existing System and Problem Description

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

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.

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.

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

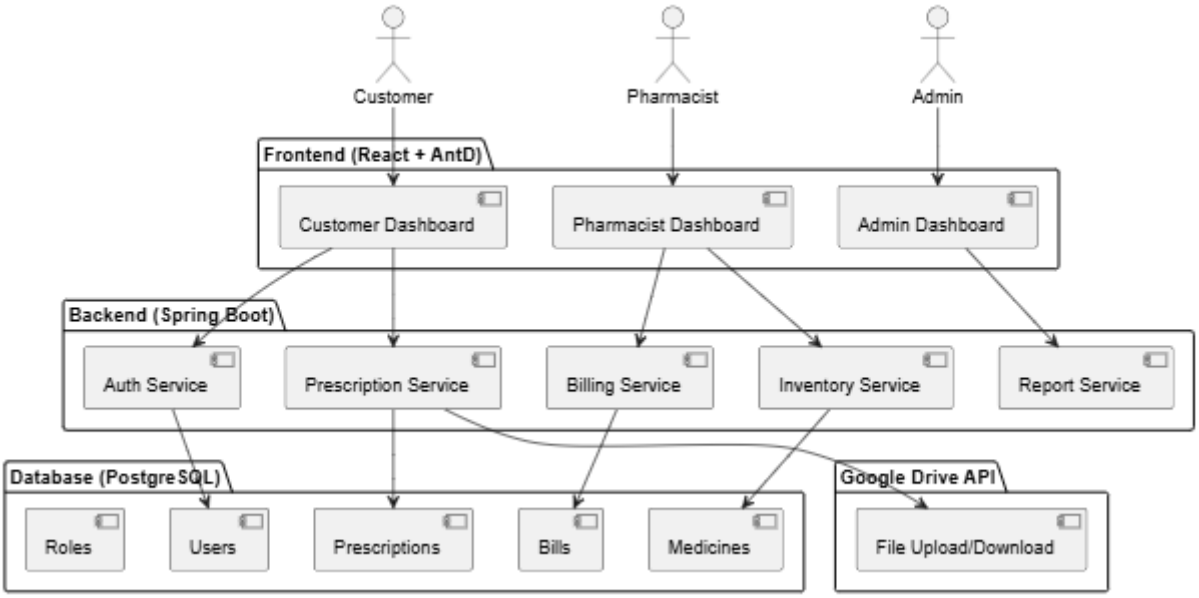
The design adheres to these core principles:

- **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.



3.4 Component and Module Design

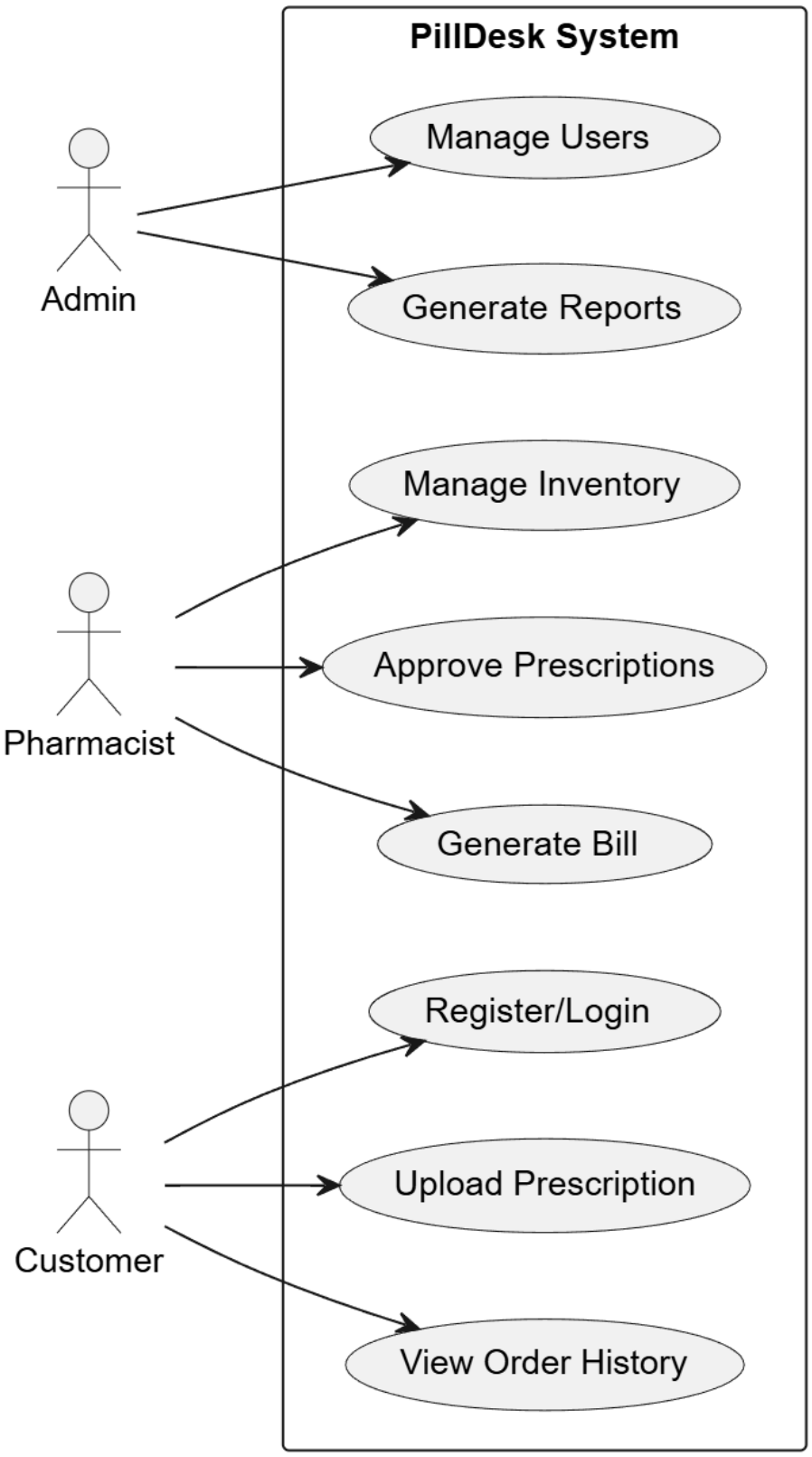
Key Modules:

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.

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

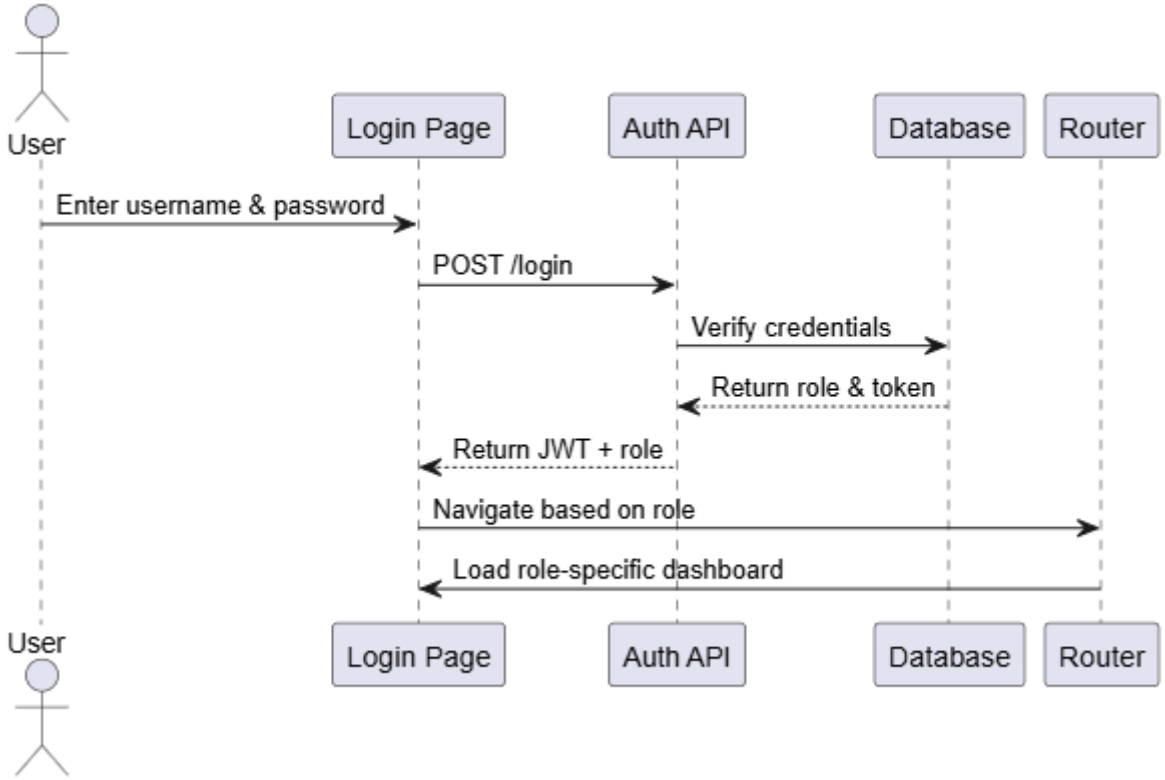
3.5 Workflow and Behavioral Modeling

Use Case Diagram

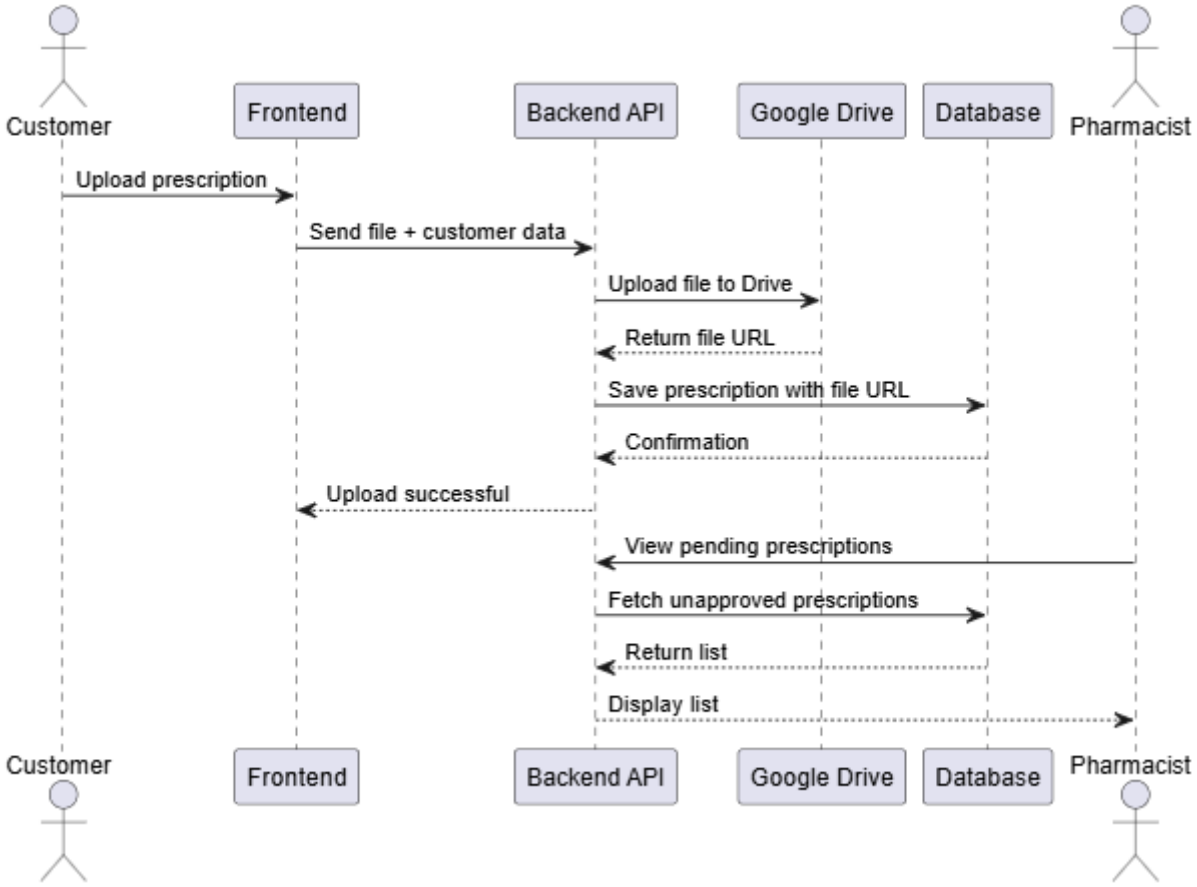


Sequence Diagrams

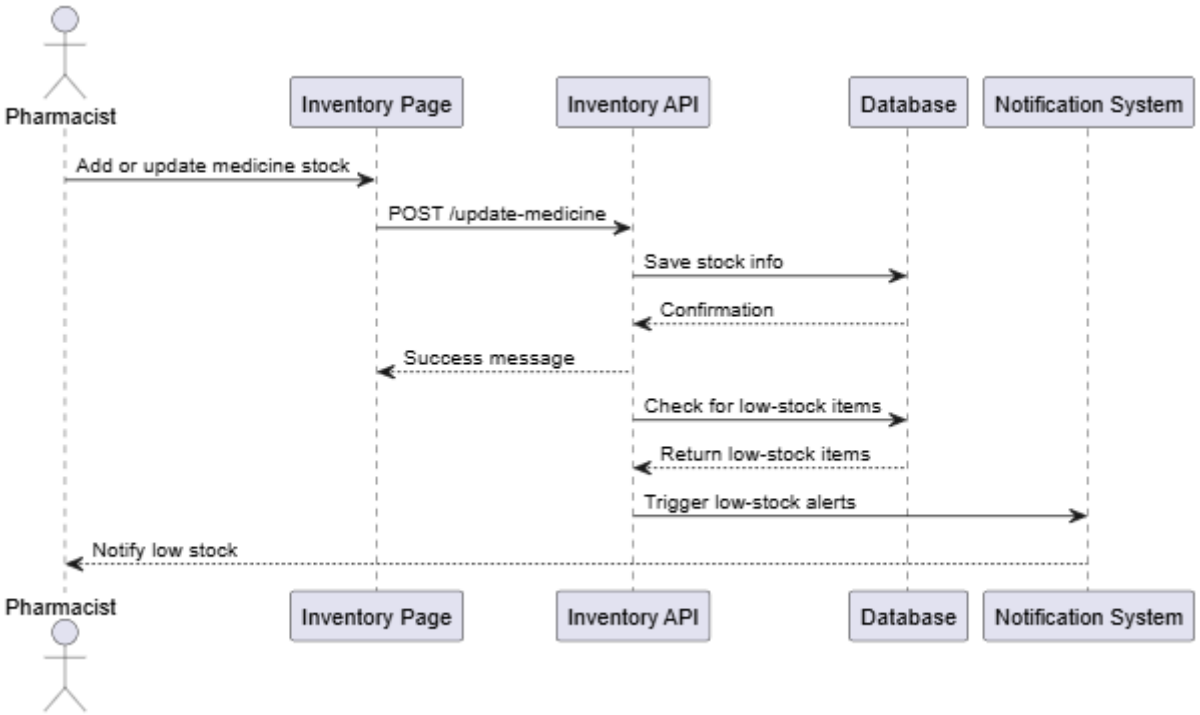
- Login and role-based redirection



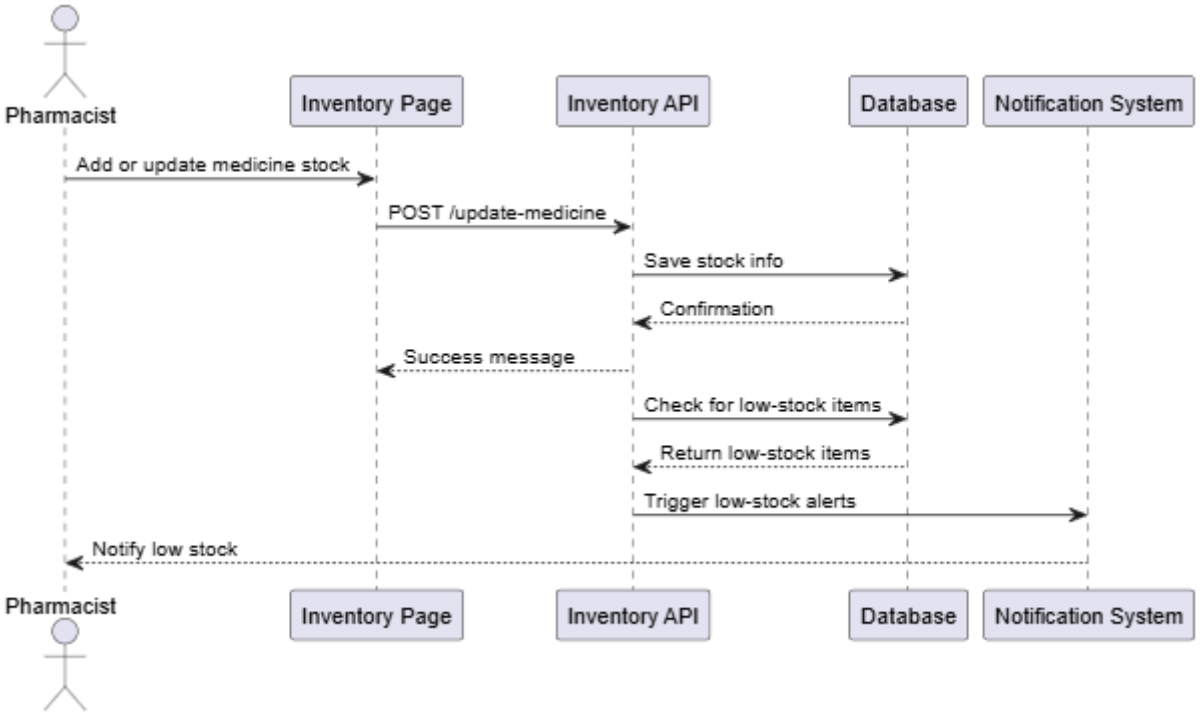
- Prescription upload and pharmacist approval



- Inventory update and low-stock alert



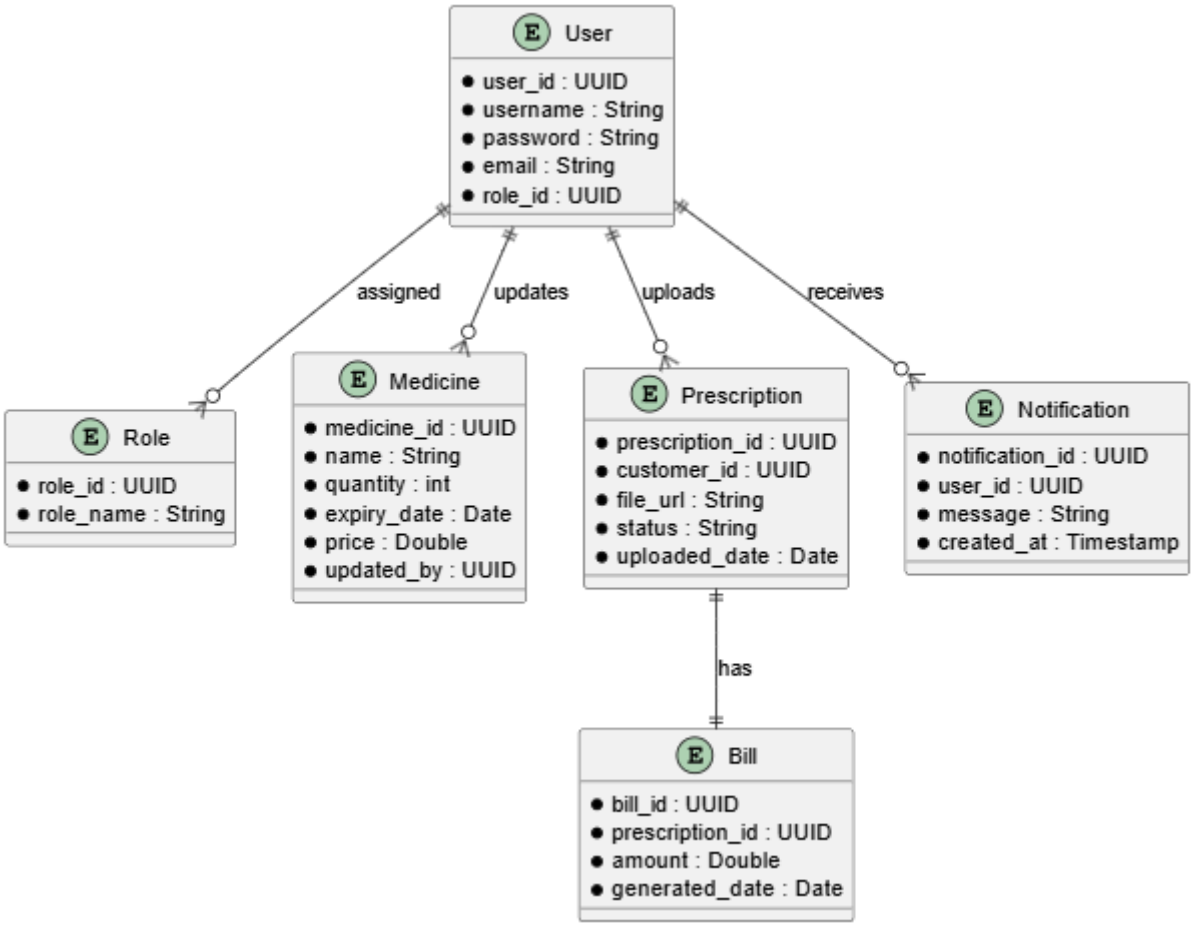
- Bill generation and report export



3.6 Data Modeling

Entity-Relationship Diagram (ERD)

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



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).
- Responsive behavior across devices.

📌 *Insert Figure 3.8: Sample UI Screens (Login, Dashboard, Upload Form)*

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

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:

properties

CopyEdit

spring.datasource.url=jdbc:postgresql://localhost:5432/pilldesk

spring.datasource.username=postgres

spring.datasource.password=your_password

jwt.secret=your_secret_key

- .env for frontend:

env

CopyEdit

VITE_API_BASE_URL=http://localhost:8080/api

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.

java

CopyEdit

```
@PostMapping("/login")
public ResponseEntity<?> authenticateUser(@RequestBody LoginRequest request) {
   // validate credentials
   // generate JWT
   return ResponseEntity.ok(new JwtResponse(token, user.getRole()));
}
```

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.

tsx

CopyEdit

```
// Frontend (React) using fetch
const uploadPrescription = async (file) => {
  const formData = new FormData();
  formData.append('file', file);
  const res = await fetch('/api/prescriptions/upload', {
    method: 'POST',
    headers: { Authorization: `Bearer ${token}` },
    body: formData,
  });
};
```

4.3.3 Inventory and Alert Logic

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

```
java
CopyEdit
if (medicine.getQuantity() < medicine.getThreshold()) {
   notificationService.notifyLowStock(medicine);
}
```

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.

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

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

Test cases were organized using test scenarios and expected results.

5.2.2 API Testing (Postman)

- All backend endpoints (e.g., /api/login, /api/medicines, /api/prescriptions/upload) were tested using Postman collections.
- Token-based authentication was validated using Postman headers.
- Edge cases such as empty inputs, unauthorized access, and malformed tokens were tested.

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

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.

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 (web is responsive though).
- Lack of a complete audit log or change tracking system.
- Notifications are limited to the UI (no email/SMS integration yet).

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

It is very important to acknowledge any of the work of others that the candidate used or adapted in the project, or that provided the essential background or context to the dissertation. Please note that IEEE is the recommended referencing and citation style for your dissertation. It is also recommended that the candidate use a reference management tool such as Mendeley or Zotero

Following examples show how the referencing should be done.

Example 1:

Systems analysis and design techniques are considered essential for developing client/server and web-centric applications [6, 7].

Example 2:

Software testing [8] is an iterative process.

Example 3:

“Plagiarism is an act of fraud. It involves both stealing someone else's work and lying about it afterward” [9].

In the **References** section, each citation should be listed in the relevant format (Refer to an IEEE referencing and citation style guide). For example, the reference section entries for the above three examples would be;

[6] J.L. Whitten and L.D. Bentley, *Systems Analysis and Design Methods,* 7th ed. Tata McGraw-Hill, 2007.

[7] UCSC, *The Virtual Learning Environment for the BIT Students,* 2006. [Online] Available: http://vle.bit.lk , [Accessed: 30 Oct, 2013]

[8] I. Sommerville, *Software Engineering*, 8th ed. Addison-Wesley, 2006.

[9] Plagiarism.org - Best Practices for Ensuring Originality in Written Work, "What is Plagiarism?", 2015. [Online]. Available: http://www.plagiarism.org/plagiarism-101/what-is-plagiarism/. [Accessed: 29- Dec- 2015].

Appendices

The appendices include further information that is not essential to be included in the main text, but nevertheless could be useful to interested readers. The following appendices should be included in the dissertation:

Appendix A - System Manual

Technical documentation is included here. These details should guide candidates who wish to continue or use your project work and allow amendments and extensions to the code. Provide program installation, compilation, and execution details. Documentation should point to locations where DLL's and reusable code can be found if they are publicly available.

Appendix B - User Manual

May include thorough and comprehensive documentation at a level, which is appropriate to the identified users. User documentation may cover all aspects of the system, with appropriate screenshots and explanations. **Failing to include such documents means that you have failed to implement a critical component of the system and it could result in not calling for project evaluation**.

Appendix C - Management Reports

In addition to producing day to day transaction reports (e.g. a payroll system should produce an individual pay sheet, coin analysis to make cash payments, EPF report, etc.) a system must produce summarised reports for the management (e.g. monthly, quarterly payments made by organisation, employees, overtime hours by employee, etc.). These reports should be included here. The usefulness of the system will be judged using these reports. **Failing to include such reports means that the candidate had failed to achieve his objectives and it could result in not calling for project evaluation**. Ensure that the reports contain meaningful information which could be obtained through your system using a sufficient amount of data.

General Guidelines

Candidates are strongly advised, while writing is in progress, to show each chapter to their supervisors for necessary feedback especially on technical content. Please follow the instructions below to minimize correction time.

- The candidates are advised to follow the typing recommendations given below to typeset their dissertations.

    - Dissertation text Times New Roman - 12pt
    - Text in tables and code listing - 11pt
    - Line spacing (preface and main text) - 1.5
    - Line spacing appendices - 1.0
    - Left margin - 37mm
    - Top/bottom/right margins - 25mm
    - Chapter heading - 24pt bold
    - Section headings - 16pt
    - Subsection headings - 14pt
    - Other headings - 12pt bold
    - Tables headings font - 11pt bold

- All pages should be numbered including Chapter 1 beginning from page 1. Use roman numerals for front matter as used in this guidelines document.

- **Plagiarism** is the presentation of another person's thoughts or words as though they were the candidate's own. The candidate should avoid this when writing his dissertation. All sentences or passages quoted in reports from other people's work have to be specially acknowledged by clear cross-referencing to author, work and page(s). Direct quotations from published or unpublished work of others should always be clearly identified as such by being placed inside quotation marks, and full reference to their source should be provided in the proper form. Equally, if another person's ideas or judgements are summarized, the candidate should refer to that person in the main text of the dissertation, and include the work referred to in the references section of the dissertation. Failure to adhere to these rules may result

in an allegation of cheating. **All suspected cheating will be reported as examination offenses**. Any illustrations, which are not the work of the candidate can be used only with the explicit permission of the originator and should be specially acknowledged. The project is an important component of the degree and plagiarism in project work is taken very seriously, and when discovered will imply severe penalties and consequences for the culprit's degree and possibly for his entire future career. Such a candidate will fail the project and the degree examination as a whole when plagiarism in project work is discovered. Candidates will not be allowed to repeat the project and other degree components for a specified number of years depending on the decision by the university. Therefore, it is important to give credit where it is due and acknowledge all work borrowed and emphasize what the candidate's distinct contribution has been in the project. **Similarity Index should be less than 20% to avoid consideration for possible plagiarism attempts.**

Writing Style

The following are some of the general points you should use them to improve writing style:

- Try to minimize mixing of unrelated issues in the same chapter. For example, in the design chapter, do not discuss testing issues. One exception to this rule would be when arguments need to justify actions related to the current topic.
- Throughout the dissertation, use standard software engineering or computer science terminology. (refer Guide to the Software Engineering Body of Knowledge Version 3.0 SWEBOK®,)
- Use a consistent naming convention. Do not refer to the same concept or "thing" by different names in different parts of the dissertation.
- Write short, focused, and coherent paragraphs. Each paragraph should have a clearly stated topic sentence that describes the main issue/context covered in the paragraph. The rest of the sentences should provide discussions on the topic sentence, or answer the question raised in the topic sentence.

- Do a grammar and style check before submission. It is ESSENTIAL that you proofread your dissertation manually before printing the final copy for submission.
- Omit needless words—redundant modifiers, pompous diction, excessive detail.
- Eliminate "narration," expressions such as "It is my opinion that," "I have concluded," "the main point supporting my view concerns," or "certainly there is little doubt as to. ". Focus attention solely on what the reader needs to know.