WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF SOFTWARE ENGINEERING

Medicine Locator System

INDUSTRIAL PROJECT PROPOSAL

BY

1. Yidnekachew Bantirga NSR/1545/12
2. Betsegaw Abebe NSR/0322/12
3. Tolasa Ganati NSR/1449/12

PROJECT ADVISOR: Mr. Bekretsion B.

WOLKITE UNIVERSITY, WOLKITE, ETHIOPIA
MAY 2024

WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF SOFTWARE ENGINEERING

Medicine Locator System

SUBMITTED TO DEPARTMENT OF SOFTWARE ENGINEERING

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR

THE DEGREE OF BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING

BY

1. Yidnekachew Bantirga NSR/1545/12
2. Betsegaw Abebe NSR/0322/12
3. Tolasa Ganati NSR/1449/12

PROJECT ADVISOR:Mr. Bekretsion B.

Wolkite University, Wolkite, Ethiopia
May 10, 2024

DECLARATION

This is to declare that this project work which is done under the supervision of Mr. Bekretsion B. and having the title Medicine Locator System is the sole contribution of:Tolasa Ganati, Yidnekachew Bantirga and Betsegaw Abebe. No part of the project work has been reproduced illegally (copy and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: _____

Group Members:

Full Name	Signature
_____	_____
_____	_____
_____	_____

Approval Form

This is to confirm that the project report entitled Medicine Locator System submitted to Wolkite University, College of Computing and Informatics Department of Software Engineering by:

Tolasa Ganati, Yidnekachew Bantirga and Betsegaw Abebe is approved for submission.

Advisor Name	Signature	Date
Department Head Name	Signature	Date
Examiner 1 Name	Signature	Date
Examiner 2 Name	Signature	Date
Examiner 3 Name	Signature	Date

ACKNOWLEDGMENT

First of all we would like to thank Almighty GOD for the strength, GOD has given us throughout our life and this project nothing could happen without the help of GOD. Secondly, we would like to express our gratitude to our advisor Mr. Bekretsion B. for his help, willingness and commitment in giving his precious time to help us to accomplish this work.We are also very grateful and would like to extend our sincere thanks to our department staff members and students of our Department Software Engineering for sharing their ideas, suggestions, and support and especially for their commitment. We really give a great respect and credit to everyone who involved in our project tasks.

CONTENTS **PAGE**

LIST OF FIGURES

List of Tables

List of Abbreviations

1) WKU ……Wolkite University
2) MERN ……..Mongo Database, Express js, React js, Node js
3) AI …….Artificial Intelligence
4) UI ……User Interface
5) OTC …….Over-the-counter
6) OOSAD ……Object Oriented System Analysis and Design
7) MLS …….Medicine Locator System
8) UML …….Unified Modeling Language
9) API ……Application Programming Interface
10) JWT……..Json Web Token
11) GPS ……..Global Positioning System
12) FAQs ……Frequently Asked Questions
13) HTTP……Hypertext Transfer Protocol

Executive summary

In our community searching and getting drugs is not in a computerized way, the way to treat the patient and manage patient information are not in digital form. The main objective of this project is to develop a computerized Medicine Locator System. The proposed system changes the existing manual system into a computerized way of working environment by identifying the existing problem and making analysis on it then finally design the new system. Currently accessing drugs is done manually, making it very labor intensive and ineffective. The system we have proposed is going through a stage of life cycles requirement gathering, requirement analyzing, system designing, implementing, testing and maintaining. In every stage we will use different techniques in order to achieve our main objective.

CHAPTER ONE

1. INTRODUCTION

In today's fast-paced world, timely access to essential medications is a significant challenge. This challenge is further amplified by factors such as geographical barriers, lack of information about medicine availability, and price discrepancies across pharmacies. These issues highlight the need for a comprehensive solution that can bridge the gap between pharmacies and consumers.

The project focuses on using digital technology to improve pharmaceutical supply chain accessibility and efficiency, ensuring reliable access to essential medications. The Medicine Locator System is an innovative digital platform aimed at connecting users with pharmacies to streamline medicine procurement. It offers features for different user types, including pharmacists, registered customers, and guests, and uses location-based services to display nearby pharmacies with the required medicine in stock.

Registered customers can manage accounts, order medicines for delivery and review pharmacies. Guests can browse pharmacies, filter drugs, and explore by categories.

Pharmacists manage profiles, view patient orders. The system offers emergency services to locate 24/7 pharmacies and stores with urgent supplies, ensuring medication access during emergencies.

By providing detailed information about each medicine and enabling price comparison across different pharmacies, the Medicine Locator System promotes informed decision-making. It enhances efficiency in the pharmaceutical supply chain and contributes to the overall well-being of the community by ensuring a reliable and accessible supply of essential medications.

1.1. Statement of the Problem

The problem at hand pertains to the inefficiency and inconvenience inherent in the current manual system for acquiring essential medications. In this prevailing system, individuals seeking specific medicines are compelled to undertake a laborious and time-consuming process. This process involves personally visiting multiple pharmacies to inquire about the availability of their required medications, giving rise to various issues including frustration, time wastage, unnecessary costs, and an unwarranted expenditure of energy. The existing system operates entirely manually, devoid of any automated or semi-automated mechanisms to facilitate the location and procurement of vital medications.

Inefficiencies in the Current System:

- ➔ Time Wastage: Visiting multiple pharmacies, waiting in queues, and repeatedly inquiring about medication availability leads to significant time inefficiency, affecting consumers and pharmacy staff alike.
- ➔ Frustration: The manual system results in considerable frustration, especially for individuals already grappling with health issues, as fruitless trips can be emotionally taxing.
- ➔ Cost Inefficiency: Traveling to various pharmacies incurs unnecessary transportation costs that accumulate over time, while multiple visits and medication delays pose financial challenges, especially for those with chronic health conditions.
- ➔ Inventory Management: Pharmacies struggle to efficiently manage and restock their inventories, leading to occasional shortages or overstocking, hindering timely access for patients.
- ➔ Data Accuracy: Manual record-keeping can result in errors in tracking medication availability, prescription details, and patient information, potentially jeopardizing patient safety and care quality.

1.2. Objective of the Project

1.2.1. General objective

The general objective of this project is to develop a digital platform that modernizes and streamlines the medication procurement process by replacing the current manual system with an efficient automated system.

1.2.2. Specific objectives

- To achieve the general objective, the following specific objectives are identified
- To gather user and stakeholder requirements to shape the system's functionality.
- To analyze and document requirements to create a clear roadmap for system development.
- To develop a system design based on gathered requirements to create a structured design plan that considers all requirements.
- To design user interfaces for nearby pharmacy exploration, drug search/filtering and accessing their details, pharmacy inventory management, etc.
- To develop an inventory management system for pharmacy
- To implement a user account management system for efficient user authentication and authorization.
- To develop a comprehensive drug search and filtering system.
- To implement a system for users to reviews and rate pharmacy to help other users ➔ To develop a medication ordering system for users to place orders for delivery or pickup.
- To perform testing by using different testing techniques.

1.3. Feasibility Analysis

1.3.1. Technical feasibility

The proposed system is technically feasible. The development team is proficient in the MERN(Mongodb, Expressjs, React Native, React and Nodejs) stack, and the required technology

tools (laptops, internet access, programming software) are readily available, ensuring a feasible implementation.

1.3.2. Operational feasibility

Operational feasibility is promising, as the system addresses real user needs, offering a user-friendly interface that streamlines medication procurement. Management support is evident, and users seek a more efficient alternative to current practices. The system is expected to significantly reduce the time required for operations, making it a practical and valuable addition to the healthcare landscape.

1.3.3. Economical feasibility

Beyond financial considerations, the system brings a host of benefits to users, such as time savings, cost reduction, increased access, scalability, and improved patient care, further underlining its value for healthcare providers and patients. This analysis underscores the system's economic potential, long-term sustainability, and the enhanced well-being of its users.

1.4. Scope and Limitation of the Project

1.4.1. Scope of the Project

The scope of this system is to create an integrated, user-friendly platform that connects consumers with pharmacies, enabling reliable access to essential medications, improving the overall medication procurement experience, and enhancing pharmacy inventory management. The system aims to address the identified inefficiencies and inconveniences in the current manual system, ultimately benefiting both consumers and pharmacy establishments.

- The system will facilitate an efficient user registration process, ensuring easy onboarding
- for all users.
- Robust user authentication and authorization mechanisms will be implemented to
- maintain data security and user privacy.
- The system will offer a user-friendly interface for searching medications by various drug
- fields.
- It will provide precise drug filtering options, allowing users to search for medications
- based on location, price, user reviews and rating.

- User ratings will be utilized to offer recommendations that guide users in selecting a
- suitable pharmacy.
- The system will enable effective drug stock management for pharmacies, helping them
- efficiently manage and restock their inventory.
- Drug order management system for pharmacy.

1.4.2. Limitations of the project

While the system offers significant enhancements to medication procurement and inventory management, it's important to note the following limitations:

- The system prohibits the ordering of prescription drugs in compliance with legal regulations and to ensure user safety and security.

1.5. Significance of the project

The Medicine Locator System holds paramount significance in the healthcare landscape. It offers numerous benefits to patients, pharmacists, and the healthcare ecosystem, revolutionizing medication access, improving efficiency, reducing costs, and enhancing patient care. This innovative solution also paves the way for future advancements in digital healthcare

- Enhanced Drug Access: The system significantly improves patient access to medications, especially for those with limited mobility or in remote areas.
- Time Efficiency: It reduces the time needed for medication procurement, ensuring timely access to medicines and improving overall efficiency.
- Cost Savings: Patients benefit from informed choices, reducing transportation costs, while pharmacists enhance their inventory management, lowering operational costs.
- Improved Patient Care: The system ensures accurate and timely medication access, reducing errors and enhancing patient safety.

- Pharmacist Empowerment: Pharmacists can streamline operations and potentially expand their business, thanks to improved order handling.
- Feedback Mechanism: Patient reviews enable continuous improvement in healthcare services.
- Digital Healthcare Advancement: The system represents an innovative approach to addressing real-world healthcare challenges through technology, setting a precedent for digital healthcare solutions.

1.6. Beneficiary of the project

The Project brings forth a range of beneficiaries. Patients experience improved medication access and reduced costs, pharmacists benefit from more efficient operations, and the healthcare system sees better patient care and cost savings. Furthermore, the project's economic viability generates revenue, ensuring sustainability and growth, making it a win-win for all involved.

- ➔ Patients: Individuals gain improved access to essential medications, reduced costs, and enhanced overall healthcare experiences.
- ➔ Pharmacists: Pharmacy professionals benefit from streamlined operations, improved efficiency, and potential business growth.
- ➔ Healthcare System: The system contributes to better patient care, reduced medication errors, and overall cost efficiency in healthcare.

1.7. Methodology of the Project

The methodology of the Medicine Locator System project will involve several stages, including requirements gathering, system analysis and design, implementation, testing, and deployment.

1.7.1. Data Collection Tools/Techniques

Data collection will be carried out using a combination of techniques such as observation, questionnaires and interviews. Observations will be made at local pharmacies to understand their operations and challenges. Questionnaires will be distributed to potential users to gather information about their needs and preferences. Interviews will be conducted with pharmacists and medical professionals to gain insights into the pharmaceutical supply chain.

1.7.2. System Analysis and Design

The system will be analyzed and designed using Object-Oriented System Analysis and Design (OOSAD). This approach will allow us to model the system components effectively, representing real-world entities such as users, pharmacies, and medicines as objects with properties and behaviors.

1.7.3 System Development Model

The project will utilize the Iterative Model, an iterative and incremental approach where development occurs through cycles or iterations. This model emphasizes flexibility and allows for continuous refinement of the software based on feedback obtained during each iteration.

1.7.3. System Testing Methodology

The system will be subjected to various levels of testing including unit testing, system testing, acceptance testing, and integration testing to ensure its functionality, reliability, performance, and ease of use.

1.7.4. Development Tools and Technologies

The Medicine Locator System will be developed using a variety of tools and technologies to ensure a robust, user-friendly, and efficient application.

Table 1.0.1 Development tools and technologies

Tools	**Activities**
Vs code	For Coding
React.js	For frontend
React Native	For mobile Application
Node.js	For Backend
MongoDb	For Backend(Database)
Google Docs	For Documentation
Google Slides	For Presentation
Draw.io.	For UML (system Architecture)
Figma	For UI design
Android Studio	For Mobile Application Development

1.11. Document Organization

Chapter one defines and describes concepts with regard to the introduction of the chapter that discusses problems in the existing system and with the objective of the proposed system how data is collected for the method.

Chapter two focuses on the current system. This is about its strength and weakness, actors participating must be followed by the actors in the existing system and what the system should do and quality characteristics of the system.

Chapter Three is about Overview of proposed system which includes functional and non-functional requirements, Hardware and software requirements and security and safety procedure.

Chapter four consists of flow of events which is scenario, use case model with its description of the major use cases. It deals with system analysis and designing that is defined by functional, object and dynamic models which means functional model described by use case diagram, object model described by class diagram and dynamic model described by sequence, activity and state chart diagram.

Chapter five deals with system design. Which includes the overview of the system, design consideration, design goal, design trade-offs, architecture of the system, subsystem decomposition, persistent data management and class interface.

CHAPTER TWO

2. DESCRIPTION OF THE EXISTING SYSTEM

2.3. Introduction of Existing System

A pharmacy is a retail establishment that is licensed to dispense prescription medications and other healthcare products. Pharmacies are typically owned and operated by pharmacists, who are healthcare professionals trained in the selection, preparation, and dispensing of medications. Pharmacists also provide medication counseling, immunizations, and other healthcare services.

The core business of a pharmacy is to dispense prescription medications. When a patient receives a prescription from a doctor, they can take it to any pharmacy to get it filled. The pharmacist will review the prescription, check for any drug interactions, and make sure that the patient is taking the correct medication and dosage. The pharmacist will also provide medication counseling to the patient to ensure that they understand how to take the medication properly and safely.

In addition to dispensing prescription medications, pharmacies also sell a wide variety of over-the-counter (OTC) medications, health and wellness products, and personal care items. Pharmacists can also provide advice on OTC medications and other health and wellness topics.

The business logic of a pharmacy is to provide medications and healthcare services to patients in a safe, efficient, and affordable manner. Pharmacies generate revenue by charging a fee for dispensing prescription medications and selling OTC medications and other products. Pharmacies also receive reimbursement from insurance companies for dispensing prescription medications.

The existing system for how drugstores work is highly regulated, with specific laws and regulations in place to ensure the safety and efficacy of medications dispensed to patients.

Obtaining a Prescription: To obtain a prescription for a medication, a patient must first see a doctor or other healthcare provider. The healthcare provider will diagnose the patient's condition and determine the appropriate medication and dosage. The healthcare provider will

then write a prescription, which is a signed order to the pharmacist to dispense the medication to the patient.

Filling the Prescription: Once the patient has obtained a prescription, they can take it to a drugstore to be filled. The pharmacist will review the prescription to ensure that it is valid and complete. The pharmacist will then fill the prescription by dispensing the correct medication and dosage to the patient.

Counseling the Patient: Before dispensing the medication, the pharmacist will counsel the patient on how to take the medication safely and effectively. This includes discussing the medication's dosage, frequency of administration, side effects, and interactions with other medications.

Dispensation: Once the patient has been counseled on the medication, the pharmacist will dispense the medication to the patient. This may involve counting pills, labeling the medication container, and providing the patient with written instructions on how to take the medication.

2.4. Users of Existing System

The key stakeholders in the pharmaceutical industry supply chain include raw material suppliers, drug manufacturers, regulatory agencies, wholesale distributors, pharmacies and pharmacy benefit managers (PBMs), healthcare providers, and patients. Each stakeholder plays a crucial role and proper coordination between them is essential. The existing system for how drugstores work is a complex one, involving a variety of stakeholders, including:**Patients:** Patients are the ultimate users of drugstores and the medications that they dispense. **Healthcare providers:** Healthcare providers, such as doctors, nurses, and physician assistants, are responsible for diagnosing patients' conditions and prescribing medications. **Pharmacists:** Pharmacists are licensed healthcare professionals who are responsible for dispensing medications to patients and counseling them on how to take them safely and effectively.

Pharmacy technicians: Pharmacy technicians are trained to assist pharmacists with dispensing medications and other tasks.

Drug manufacturers: Drug manufacturers are responsible for developing and manufacturing prescription medications.

Insurance companies: Insurance companies provide financial coverage for prescription medications to their members.

Government agencies: Government agencies, such as the Food and Drug Administration (FDA), are responsible for regulating the safety and efficacy of medications.

2.5. Major Functions of the Existing System

The main function of pharmacies is to dispense medications and provide patient care. This includes:

Dispensing medications: Pharmacists fill prescriptions written by doctors and other healthcare providers. They also dispense over-the-counter medications and other healthcare products.

Providing patient counseling: Pharmacists educate patients about their medications, including how to take them, what side effects to watch for, and how to interact with other medications.

Managing medication therapy: Pharmacists work with patients to develop and manage their medication therapy plans. This includes things like reviewing medications for safety and efficacy, and adjusting medications as needed.

Providing other healthcare services: Pharmacies are increasingly offering other healthcare services, such as immunizations, chronic disease management, and mental health services. Pharmacies play an essential role in the healthcare system. They are responsible for ensuring that patients have access to the medications they need and that they are taking them safely and effectively.

In addition to the above, pharmacies also have a number of other important functions, such as:

Promoting public health: Pharmacies can help to promote public health by providing education and resources on topics such as immunizations, smoking cessation, and chronic disease management.

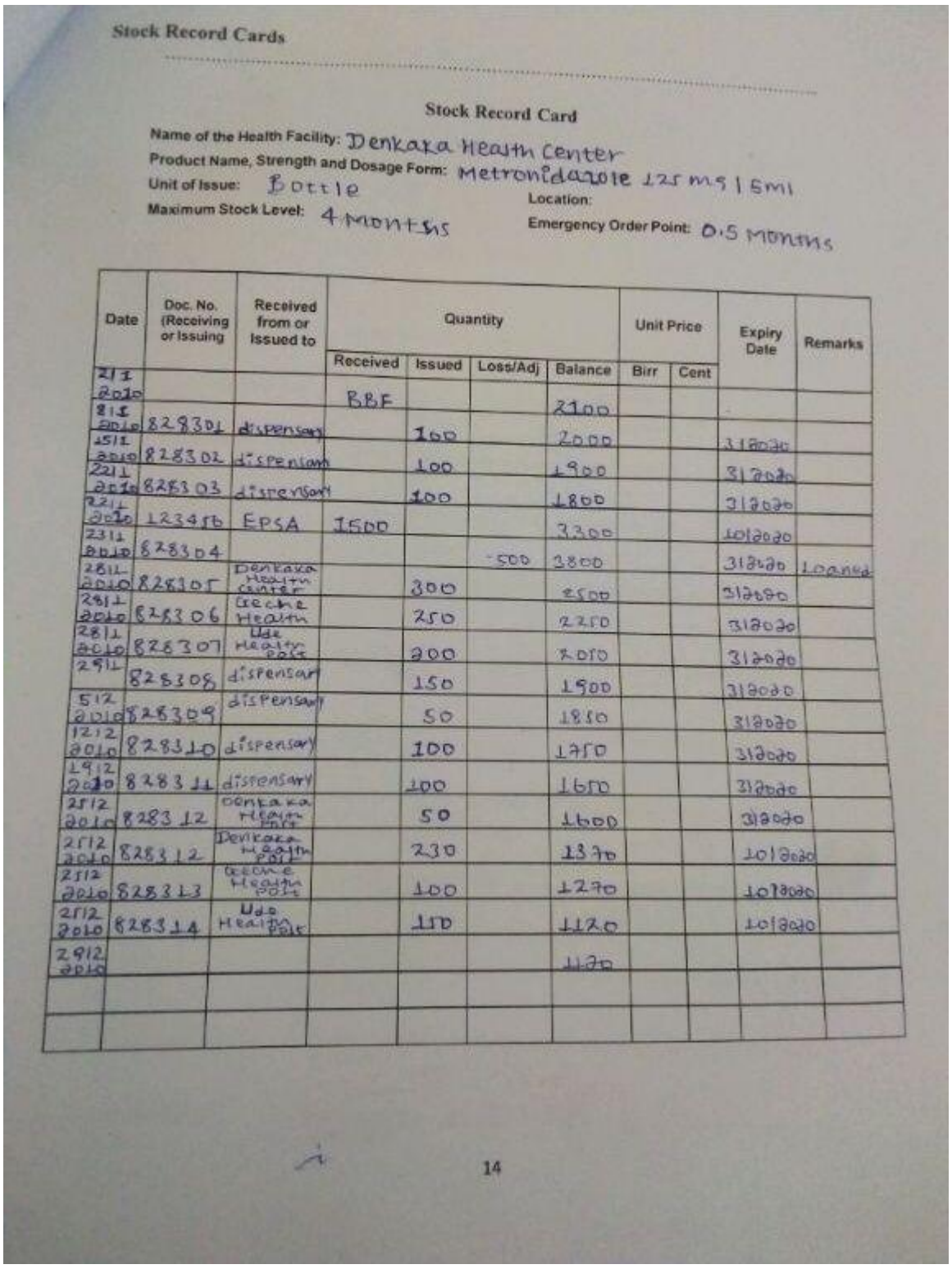
Supporting other healthcare professionals: Pharmacists work closely with other healthcare professionals, such as doctors and nurses. They provide information and support on medication use and other aspects of patient care.

Preventing medication errors: Pharmacists play a vital role in preventing medication errors. They review prescriptions carefully and check for potential interactions with other medications.

Pharmacies are essential partners in healthcare. They play a vital role in ensuring that patients have access to the medications they need and that they are taking them safely and effectively.

2.6. Forms and Other Documents of the Existing Systems

An inventory bin card is a record-keeping tool that tracks the quantity of a specific item in a specific storage location. It is used to keep a running tally of the items in stock, as well as the dates and quantities of items that have been received, issued, or transferred. Bin cards provide a record of inventory levels, which can help prevent stockouts and overstocking. Bin cards help reduce counting errors and assist during the inventory count.



Stock Record Cards

Stock Record Card

Name of the Health Facility: Denkara Health Center
Product Name, Strength and Dosage Form: Metronidazole 125 mg | 5ml
Unit of Issue: Bottle
Location:
Maximum Stock Level: 4 Months
Emergency Order Point: 0.5 Months

Date	Doc. No. (Receiving or Issuing	Received from or Issued to	Quantity				Unit Price		Expiry Date	Remarks
			Received	Issued	Loss/Adj	Balance	Birr	Cent		
2/1/2010			BBF			2100				
8/1/2010	828301	dispensary		100		2000			31/2020	
15/1/2010	828302	dispensary		100		1900			31/2020	
22/1/2010	828303	dispensary		100		1800			31/2020	
22/1/2010	123456	EPSA	1500			3300			10/2020	
23/1/2010	828304				500	3800			31/2020	Loaned
28/1/2010	828305	Denkara Health center		300		3500			31/2020	
29/1/2010	828306	Geche Health		250		3250			31/2020	
28/1/2010	828307	Udo Health Post		200		3050			31/2020	
29/1/2010	828308	dispensary		150		1900			31/2020	
5/2/2010	828309	dispensary		50		1850			31/2020	
12/2/2010	828310	dispensary		100		1750			31/2020	
19/2/2010	828311	dispensary		100		1650			31/2020	
25/2/2010	828312	Denkara Health Post		50		1600			31/2020	
25/2/2010	828312	Denkara Health Post		230		1370			10/2020	
25/2/2010	828313	Geche Health Post		100		1270			10/2020	
25/2/2010	828314	Udo Health Post		150		1120			10/2020	
29/2/2010						1120				

14

Fig 2.0.1 Bin card

2.7. Drawbacks of the Existing System

The pharmaceutical supply chain poses risks and challenges to both providers and consumers. But in the context of a health-conscious society, managing pharmaceutical supply chains presents several complexities because it involves many key components and stakeholders to efficiently deliver life-saving medicines that are vital to patients. Some of the drawbacks are:-

Long wait times: Pharmacies are often busy, and patients may have to wait a long time to have their prescriptions filled or to speak to a pharmacist. A patient may have to travel long distances to find a pharmacy that carries their medication.A patient may have to wait for hours to have their prescription filled, especially if it is a specialty medication.

Difficulty finding medications: Patients may have difficulty finding medications that are not in stock at their local pharmacy, or that are covered by their insurance plan.

Lack of communication: Pharmacists may not always communicate effectively with patients about their medications, including how to take them and what side effects to watch for. A patient may have difficulty understanding the instructions from the pharmacist about how to take their medication. A patient may have questions about their medication, but the pharmacist is too busy to answer them.

Lack of convenience: Pharmacies may not be conveniently located for all patients, or they may have limited hours of operation. A patient may have to travel long distances to find a pharmacy that carries their medication.

Technology challenges: Pharmacies need to invest in new technologies to keep up with the changing landscape of healthcare. This can be expensive, especially for independent pharmacies.

Competition from other retailers: Pharmacies are facing increasing competition from other retailers, such as grocery stores and online pharmacies. This competition can make it difficult for pharmacies to attract and retain customers.

2.8. Business Rules of the Existing System

Rule 1 : All pharmacies must be licensed by the appropriate regulatory body.

Rule 2 : Pharmacies must be staffed by licensed pharmacists at all times.

Rule 3 : Pharmacists must have a valid prescription to dispense any prescription medication.

Rule 4 : Pharmacies must store medications in a safe and secure manner.

Rule 5 : Pharmacies must keep accurate records of all medications dispensed.

Rule 6 : Pharmacists must provide counseling to patients on the proper use of medications.

Rule 7 : Pharmacists must verify the authenticity of all prescriptions.

Rule 8 : Pharmacists must check for potential drug interactions before dispensing any medication.

Rule 9 : Pharmacists must dispense medications in the correct dosage and form.

Rule 10 : Pharmacists must provide patients with written instructions on how to take their medication.

Rule 11 : Pharmacists must make sure that OTC medications are appropriate for the patient's needs.

Rule 12 : Pharmacists must provide patients with counseling on the proper use of OTC medications.

Rule 13 : Pharmacists must warn patients about potential side effects of OTC medications.

Rule 14 : Pharmacies must protect patient information in accordance with regulations.

CHAPTER THREE

3. PROPOSED SYSTEM

The proposed system, the Medicine Locator System, is a digital platform designed to revolutionize the way individuals access essential medications. It aims to address the inefficiencies and inconveniences of the current manual medication procurement process. This system connects consumers with nearby pharmacies, allowing for reliable and timely access to vital medications. Key features include efficient user registration, precise medication search and filtering and ordering, pharmacy recommendations, effective pharmacy inventory management, and drug order management. While offering significant enhancements to medication procurement and inventory management, the system does have certain limitations, such as lacking support for AI-driven symptom-based medicine recommendations. Nevertheless, it provides a promising solution to improve the overall medication procurement experience and benefit both consumers and pharmacy establishments.

3.3. Functional Requirements

3.3.1. Efficient User Registration and Authentication:

- ➔ The system shall provide a user registration feature that requires users to input their personal information, including full name, email address, and password.
- ➔ Users shall receive a verification code via code to confirm their registration and validity.
- ➔ The system should enable all users to easily log in and log out.
- ➔ The system should enable users to reset their password by sending a one-time password if they have forgotten their password.

3.3.2. Admin Approval of Pharmacies and Feedback management:

- ➔ The system administrator shall have the capability to approve or reject pharmacy registration requests.
- ➔ Approved pharmacies shall be granted access to the system's features.

- ➔ Admin should have the capability to view feedback in detail and respond.
- ➔ Admin should have the capability to access all transaction details.

3.3.3. Drug Search and Filtering :

- ➔ The system shall provide a user-friendly interface for searching medications, with options to search based on drug name, or other relevant drug fields.
- ➔ The search results shall display medications that match the user's query.

- ➔ The system should display the shortest path to the selected nearby pharmacy using Google Maps.
- ➔ Users shall be able to filter drugs and pharmacies based on:
  - ◆ Location (e.g., within a specified radius from the user's current location).
  - ◆ Price range.
  - ◆ Drug categories (e.g., over-the-counter, prescription).
  - ◆ User ratings.

3.3.4. Drug Details and Ordering:

- ➔ Users shall have the option to view detailed information about a selected medication, including dosage, usage instructions, side effects, and expiration date.
- ➔ Users shall have the option to pay for ordered medications, including choosing delivery service.

3.3.5. Pharmacy Recommendations:

- ➔ The system shall use user ratings to generate recommendations for users, suggesting suitable pharmacies based on their preferences.
- ➔ The system shall determine the user's current location through geolocation services.
- ➔ The system shall display nearby pharmacies, along with relevant details such as distance and operating hours.

3.3.6. Pharmacy Inventory Management:

- ➔ Pharmacists shall have access to a dedicated portal for managing their drug store inventory.
- ➔ Pharmacists should be able to easily update their pharmacy information.
- ➔ The system shall allow pharmacists to add, edit, or remove medications from their inventory.
- ➔ The system shall send notifications to pharmacists when the quantity of a certain medication in their inventory falls below a predefined minimum stock level.
- ➔ Notifications shall include details of the medication and a restocking recommendation.
- ➔ The system shall provide analytics to pharmacists, including the total number of drugs sold per day, month, or year.
- ➔ Pharmacists shall have access to reports and graphs to analyze their drug store's performance.

3.3.7. Drug Order management:

- ➔ The system shall maintain a list of drug orders made by users.
- ➔ Pharmacists shall have access to this list, which includes order details, customer information, and order status.
- ➔ Pharmacists shall have the capability to view detailed information about each order, including the prescribed medication and dosage.
- ➔ Pharmacists shall have the capability to reject or accept a drug order.
- ➔ The system should automatically refund the user's payment if their order is rejected or not delivered on time..
- ➔ Customers should have the capability to request a refund if their order is not delivered on time.
- ➔ Customers are required to confirm the delivery of their order upon its timely arrival.

➔ The system should automatically transfer the order and delivery fee to the pharmacy once the user confirms the delivery.

3.4 Non-functional Requirements

Non-functional requirements for a Medicine Locator System define the system's qualities, constraints, and performance characteristics that go beyond its core functionality. These requirements are essential to ensure that the system operates effectively and efficiently while meeting user expectations.

Here are non-functional requirements for a Medicine Locator System:

➔ **Performance:**

- ◆ Response Time: The system should provide fast response times, ensuring that users can quickly find nearby pharmacies and medicine availability information. The target response time for search queries should be within 2 seconds, aiming for an optimal user experience.
- ◆ Scalability: The system should be scalable to accommodate an increasing number of users and pharmacies without significant degradation in performance.
- ◆ Concurrent Users: It should support a large number of concurrent users without a noticeable slowdown.

➔ **Availability:**

- ◆ Uptime: The system should maintain an exceptional level of availability, with minimal planned or unplanned downtime, to ensure uninterrupted access for users at all times.

➔ **Security:**

- ◆ Data Security: Ensure that user data, especially sensitive health information, is stored securely, and access is restricted to authorized personnel.

- Authentication and Authorization: Implement strong user authentication and authorization mechanisms to prevent unauthorized access to the system.
- Error Handling: The system should gracefully handle errors, providing informative error messages to users and logging errors for troubleshooting.
- Data Integrity: Ensure the integrity of location and medicine data, with regular data validation and verification.

➔ **Scalability:**

- Data Storage: The system should be able to handle a growing database of pharmacy locations and medicine availability information.
- User Base: Scalability is crucial as the user base grows, with the ability to accommodate a larger number of users and data.

➔ **Usability:**

- User Interface: The user interface should be intuitive, user-friendly, and accessible to people of various skill levels and abilities.
- Accessibility: Ensure that the system complies with accessibility standards.

➔ **Compatibility:**

- Cross-Platform: The system should work seamlessly on various platforms, including web browsers, mobile devices, and different operating systems.
- Browser Compatibility: Ensure compatibility with commonly used web browsers.

➔ **Compliance:**

- Regulatory Compliance: Ensure that the system complies with relevant laws and regulations related to healthcare, data privacy, and any other applicable regulations in the region where it is deployed.

➔ **Performance Testing:**

- Regularly conduct performance testing to ensure that the system can handle expected loads and user interactions efficiently.

➔ **Cost Constraints:**

- ◆ Adhere to budgetary constraints and optimize resource utilization to ensure cost-effectiveness.

➔ **Integration:**

- ◆ Support integration with external systems and APIs for accessing real-time medicine availability and location data.

These non-functional requirements help ensure that the Medicine Locator System not only functions correctly but also performs well, is secure, and meets the expectations and constraints of both users and regulatory authorities.

3.4.1. User Interface and Human Factors

The interface for a medicine locator system should be designed to be user-friendly and intuitive, catering to a diverse range of users with varying levels of expertise. To determine the appropriate interface, it's important to consider the following aspects:

User Expertise:

- General Public: Many users of a medicine locator system may have minimal technical expertise. The interface should be designed with simplicity in mind, with clear, straightforward navigation.
- Healthcare Professionals: Some users, such as pharmacists or healthcare providers, may have a higher level of expertise. The system should provide advanced search and filter options to accommodate their specific needs.

User Segmentation:

- Consider segmenting users based on their roles and needs. For example, patients might search for nearby pharmacies, while healthcare providers may need to locate specific medications or check stock availability.

Search and Navigation:

- Provide a search bar for users to enter medication names, pharmacy

names,and location of pharmacy.

Results Display:

- Display results in a user-friendly format, with relevant information such as pharmacy names, addresses, contact information, medication availability, and distance from the user's location.
- Use maps and geolocation to visually show pharmacy locations in relation to the user's current location.

Filtering and Sorting:

- Implement filters and sorting options to help users refine their search results. Users should be able to filter by criteria like distance, open hours, in-stock medications, and more.

Feedback and Ratings:

- Allow users to leave feedback and ratings for pharmacies or specific medications to help others make informed decisions.

Mobile Responsiveness:

- Ensure that the interface is responsive and works well on both desktop and mobile devices. Many users will access the system from smartphones.

User Support and Help:

- Provide clear instructions, tooltips, and a help section to assist users in using the system effectively. Include contact information for customer support if users encounter issues.

Feedback Mechanism:

- Allow users to report inaccuracies or issues with pharmacy information. This can help maintain the system's accuracy.

Continuous Improvement:

- Gather user feedback and conduct usability testing regularly to identify areas for improvement in the interface and overall user experience.

3.4.2. Hardware Consideration

The hardware compatibility requirements for a medicine locator system may depend on various factors, including the platform or devices on which the system is intended to run.

Additionally, the system's interaction with other hardware systems can vary based on its specific functionality.

Hardware Compatibility Requirements:

Server Hardware: The server that hosts the system (e.g., a web server or cloud-based server) should meet specific hardware requirements to ensure optimal performance. This includes considerations such as CPU, memory, and storage capacity.

Client Devices: Consider the hardware requirements for the devices on which users will access the system. This can include desktop computers, laptops, tablets, and smartphones.

Geolocation Services: Because the system relies on geolocation services to determine a user's location or the location of pharmacies, it should be compatible with devices that support GPS or other location-based technologies.

Communication Hardware: The system may rely on communication hardware (e.g Internet connectivity) to communicate with external systems and services.

Table 3. 1Hardware compatibility Requirement

users	**Item**	**Specification**
Pharmacies	Desktop/Laptop	Windows, macOS, or Linux-based system
	Processor	2GHZ

	RAM	512MB
	System Type	32 bit OS
customer	Smartphones/Tablets	iOS/Android
	RAM	2 GB RAM

3.4.3. Security Issues

Medicine locator system should be protected against both internal and external intrusions, as well as unauthorized access. The level of protection required should be commensurate with the sensitivity of the data and the potential consequences of a security breach. Here are some considerations for securing the system:

For External Intrusions:

- Firewalls: Implement a firewall to control incoming and outgoing network traffic. Configure it to block unauthorized access and protect the system from external threats.
- Security Certificates: Use SSL/TLS to secure data in transit. Ensure that data transmitted between the client and server is encrypted.
- Security Best Practices: Follow industry-standard security best practices for web applications, such as input validation, protection against Mongo db injection, and Cross-Site Scripting (XSS) prevention.
- Regular Security Audits: Conduct regular security audits and vulnerability assessments to identify and mitigate external threats.

For Internal Intrusions:

- Access Control: Implement access control mechanisms to restrict internal users' access to data and functionality based on their roles and permissions.
- Authentication: Require strong user authentication to ensure that only

authorized personnel can access sensitive data or perform critical actions.

- Data Encryption: Encrypt sensitive data at rest to protect it from unauthorized access by internal users who may have access to the server.

Protection Against Unauthorized Access:

- Password Policies: Enforce strong password policies and encourage users to create unique and secure passwords.
- Role-Based Access Control: Implement role-based access control (RBAC) to ensure that users only have access to the features and data that are relevant to their roles.
- Session Management: Implement secure session management to prevent session hijacking and unauthorized access to user accounts.

The system's security is fortified through the utilization of the JWT (JSON Web Token) security algorithm.

3.4.4. Performance Consideration

Responsiveness:

- Real-Time Updates: The system should be highly responsive, particularly in critical functions like locating nearby pharmacies and checking medication availability. Users expect real-time or near-real-time updates.
- User Experience: Responsiveness is crucial to providing a positive user experience. Users should not experience significant delays or slowdowns when interacting with the system.
- Optimized Frontend: Implement efficient frontend code and optimize the user interface to ensure fast loading times and smooth interactions.
- Scalability: Design the system with scalability in mind to handle increased traffic during peak usage periods without compromising responsiveness.

Concurrent User Support:

- Scalability: The system should be designed to scale horizontally to accommodate additional users as needed. This may involve load balancing and server clustering.
- Performance Testing: Conduct performance testing to determine the system's ability to handle concurrent users. Load testing tools can simulate user traffic to assess the system's performance under various conditions.

3.4.5. Error Handling and Validation

Handling exceptions is a critical aspect of designing a robust and reliable medicine locator system. Exceptions can occur due to a variety of reasons, including network issues, user input errors, or unexpected system behavior. Here's how the system should handle exceptions and the types of exceptions it should be prepared to address:

Exception Handling:

- Graceful Error Messages: Provide clear and user-friendly error messages to inform users about issues while using the system. These messages should avoid revealing sensitive system information but should guide users on how to resolve or report the problem.
- Error Pages: Create custom error pages for different types of exceptions (e.g., 404 for pages not found, 500 for internal server errors). These error pages should provide information and guidance to users and administrators.
- User Notifications: Notify users when exceptions occur, especially if it affects their current task or data submission. User notifications can help manage user expectations and prevent frustration.

Types of Exceptions to Handle:

- Validation Errors: Validate user input and handle validation errors gracefully. Provide feedback to users about incorrect or incomplete input.

- Data Retrieval Errors: Handle exceptions related to database or external data source access. Implement proper error handling to prevent data corruption and loss.
- Security Exceptions: Address security-related exceptions, such as authentication or authorization failures, by providing appropriate access control and user authentication mechanisms.
- Third-Party Service Failures: Be prepared for failures or disruptions in third-party services that your system relies on, such as geolocation services or pharmacy databases.

3.4.6. Quality Issues

The reliability, availability, and robustness of a medicine locator system should align with client and user expectations.

Reliability:The system aims for high reliability, effective error handling and exception management.

Availability:- The system is available for 24/7 access.

Robustness:- The system handles errors and unexpected situations gracefully.

Client Involvement:

- Clients participate in acceptance testing, provide feedback, and verify system functionality.
- Ongoing feedback and user input help identify areas for improvement.
- Review key performance metrics like uptime, response times, and error rates for quality assessment.

3.4.7. Backup and Recovery

- To take backup we use automated tools to regularly back up critical data, including databases, files, and configurations.

- Backups scheduled based on data change rates and criticality (e.g., daily, weekly).
- Backups stored both on-site (local server) and off-site (remote data center or cloud storage) for redundancy and disaster recovery.

3.4.9. Resource Issues

Resource constraints in a medicine locator system, impacting performance, scalability, and efficiency, include:

Server Capacity: The hosting server's processing power, memory, and bandwidth may limit concurrent user handling.

Database Storage: Storage constraints in the database, holding vital medicine information, may lead to performance challenges as data volumes grow.

Network Bandwidth: Limited bandwidth can affect system performance, especially during data transfers between server, client devices, or different components.

Geolocation Services: Reliance on external geolocation services may introduce constraints, impacting the accuracy and speed of location-based queries.

To address these constraints needs a thoughtful approach to system design, regular monitoring, and strategic allocation of resources.

3.4.10. Documentation

The level of documentation required for Medicine Locator System should encompass various types to ensure its successful development, maintenance, and use.

These typically include:

User Documentation:

- User documentation is essential to guide end-users on how to interact with the system effectively. It should be clear, concise, and user-friendly. This may include user manuals, help guides, and FAQs.

  Technical Documentation for Maintainers:
- Technical documentation is crucial for the individuals responsible for maintaining and supporting the system, such as system administrators, IT staff, and developers. This documentation should be comprehensive and cover technical aspects of the system Development Process Documentation:
- Documenting the development process is crucial for transparency, collaboration, and knowledge transfer among the development team.

CHAPTER FOUR

4. SYSTEM ANALYSIS

In this dynamic world, the field of System Analysis plays a crucial role in developing software solutions for various industries, including the healthcare sector. Our focus is on implementing a Medicine Locator System, utilizing System Analysis techniques to enhance the efficiency of the system and ensure its components work seamlessly to achieve their intended objectives. This chapter will elaborate on the System Analysis applied to the Medicine Locator System, including the design of use case diagrams, use case descriptions for identified use case, an object model class diagram, sequence diagram, activity diagram, and statechart diagram specific to the context of a Medicine Locator System.

4.3. System Model

In the ever-evolving landscape of technological solutions, the analysis model becomes a critical tool in developing systems that meet diverse needs. Whether in healthcare or other sectors, the approach of System Analysis, aiming for correctness, completeness, and consistency, is essential. This model typically encompasses three core elements: the functional model, portraying system functionality through use case diagrams; the object model, illustrating system structure with attributes, objects, and associations in class diagrams; and dynamic models, capturing internal behaviors, execution sequences, and states using sequence, state chart, and activity diagrams. This comprehensive methodology ensures a thorough understanding of a system's requirements, structure, and behavior, facilitating its development to meet the dynamic demands of various industries and users.

4.3.1. Use Case Model

The Use Case Model describes the functionality of the Medicine Locator System. It represents a discrete unit of the interaction of a user (actors of the system) with the Medicine Locator System**.** Use Case has a

description that describes the functionality that will be built in the proposed system. Use Cases are typically related to '**actors**', that is a human or machine entity that interacts with the system to perform meaningful work.

UseCase Diagram

We have used use case diagrams in order to capture the functional requirements of the system. It is the functionality of the system or the service provided by the system. In our use case diagram, we have considered the following elements.

Use Case: This is the functionality of our Medicine Locator System which has a direct interaction with the user.

Actor: any entity that interacts with our system.

Relationship: A relation between the use cases.

We identified both actors and use cases (functionalities) that exist in the Medicine Locator System as the following:

Actors Guest user

- ☐ Customer
- ☐ Super Admin:
- ☐ Admin:
- ☐ Pharmacist
- ☐ **The major Actors of the proposed system**

Guest user:- The person using the Medicine Locator System to search for medicines, view pharmacy information, and get directions to nearby pharmacies.

Customer:- The person using the Medicine Locator System to search for medicines, view pharmacy information, get directions to nearby pharmacies and order drugs.

Super Admin:-The individual responsible for managing the operations, configurations, overseeing and managing administrators within the system.

Admin:- The individual tasked with managing the operations and configurations of the Medicine Locator System.

Pharmacist:- The individual responsible for managing order processing and maintaining inventory information within the Medicine Locator System.

List of Use cases found from the functional Requirements.

- ➢ Use cases of the guest user
  - ★ Search for medicines.
  - ★ View pharmacy details.
  - ★ Register to the system.

- ➢ Use cases of the customer ★ Log in or out the system ★ Search for medicines.
  - ★ View pharmacy details.
  - ★ Get directions to a selected pharmacy.
  - ★ Rate the pharmacy.
- ➢ Use cases of System Super Administrator ★ Log in or out the system.
  - ★ Manage the Administrator.

- ➢ Use cases of System Administrator ★ Log in or out the system ★ Approve pharmacy.
  - ★ Manage the system.
- ➢ Use cases of pharmacist ★ Log in or out the system ★ Register the pharmacy.
  - ★ Update inventory information.

- ★ Manage orders.
- ★ Add or update pharmacy details.

These use cases and actors help define the functional requirements of the Medicine Locator System, outlining the interactions between users and pharmacies for efficient medicine location and acquisition.

4.3.1.1. Use Case Diagram

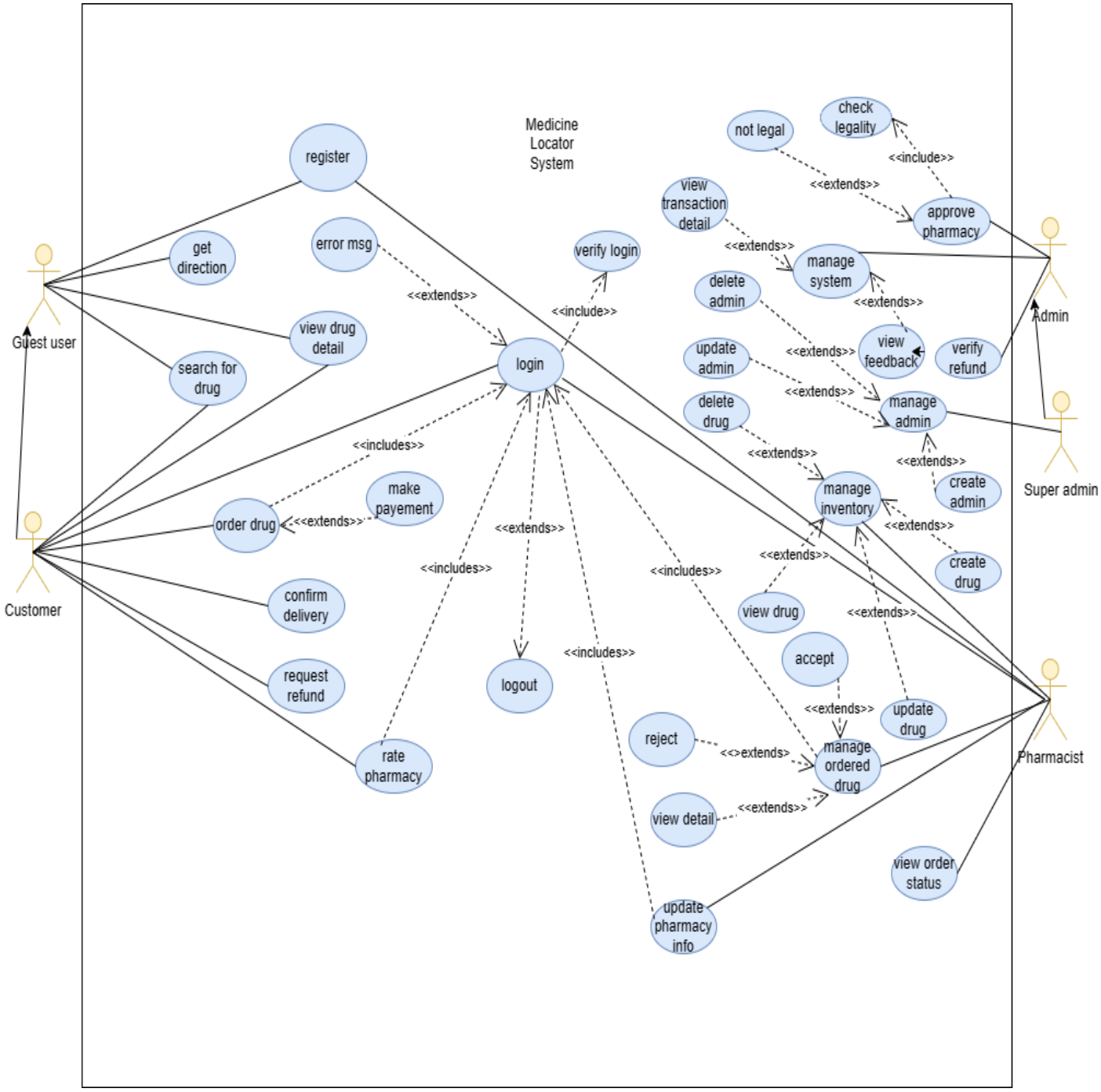


Fig 4.0.1 use case diagram

4.3.1.2. Use Case Description

Login

Table 4.1 Login Use Case Description	
case Name	Login
Actors	Customer, Super Admin, Admin,Pharmacy
Description	Admin,Super Admin, Pharmacy and any user who wants to order drugs must first log in and be authenticated and authorized.
Preconditions	The user has a valid account in the system.
Flow of event	1. The user navigates to the system's login page. 2. The user enters their username and password in the login fields. 3. The system authenticates the user's credentials. 4. If the user's credentials are valid, the system logs the user in and redirects them to the appropriate dashboard based on their user role. 5. If the user's credentials are invalid, the system displays an error message and prompts the user to try again.
Alternative Flow	2.1. If the user forgets their password, they can click on the "Forgot password?" link to reset their password.
Post conditions	The user is logged in to the system and has access to the appropriate features based on their user role.

Table 4. 1

Registration

Table 4.2 Registration use case description	
Use case name	User Registration
Actor	Customer, Admin, Pharmacy
Description	This use case describes how a user registers for an account in the system.
Preconditions	The user has accessed the registration page of the system.
Flow of events	1. The system provides a registration form where users are required to input their personal information, including full name, email address, and password. 2. After filling out the form, the user submits the registration request. 3. Upon successful submission, the system sends a verification email to the email address provided by the user. This email contains a link or code to confirm the user's registration. 4. The user confirms their registration by clicking on the verification link or inputting the verification code in the system.
Alternative flow	If there is an error during registration or email verification, the system will notify the user.
Postcondition	The user has successfully registered and verified their account.

Order management

Table 4.3 order management use case description	
Use case name	Manage orders

Actors	pharmacist
Description	Pharmacists can view and reject drug orders made by users.
Preconditions	• The pharmacist is logged into the pharmacy inventory management system. • The system has a list of drug orders made by users.
Table 4.3 order management use case description	
Use case name	Manage orders
Flow of event	1. The pharmacist selects the "Manage orders" option from the system menu. 2. The system displays a list of all drug orders made by users. 3. The pharmacist selects the order that they want to view. 4. The system displays the order details, including the customer information, order status, medication and dosage. 5. If the pharmacist determines that the order details are invalid, they can reject the order by clicking on the "Reject Order" button. 6. The system updates the order status to "Rejected" and sends a notification to the customer about the order rejection.
Alternative Flows	5.1. If the order is valid the pharmacist accepts the order by clicking the "Accept Order" button. 5.2. The system updates the order status to "Accepted" and sends a notification to the customer about the order rejection.

Post Conditions	• The pharmacist has viewed the detailed information about the selected order, including the medication and dosage. • The pharmacist has rejected the order if the order is invalid. • A notification has been sent to the customer about the order.

Pharmacy Approval

Table 4.4 pharmacy approval use case description	
Use case name	Pharmacy Approval
Actors	Admin
Description	The system administrator reviews the pharmacy's information and verifies that the pharmacy is licensed and in good standing.
Precondition	The system administrator is logged in to the system.
Flow of events	1. The system administrator views a list of pending pharmacy registration requests. 2. The system administrator selects a pharmacy registration request to review. 3. The system administrator reviews the pharmacy's information, including the pharmacy's name, address, and license number. 4. The system administrator approves or rejects the pharmacy registration request. 5. If the system administrator approves the pharmacy registration request, the system grants the pharmacy access to the system's features. 6. If the system administrator rejects the pharmacy registration request, the system denies the pharmacy access to the system's features.

	7. The system administrator sends an email notification to the pharmacy informing them of the decision.
Postconditions	The pharmacy registration request has been approved or rejected, and the pharmacy has been granted or denied access to the system's features, accordingly.

Pharmacy Inventory Management

Table 4.5 pharmacy inventory management use case description	
Use case name	Pharmacy Inventory Management
Actors	Pharmacy
Description	Pharmacists use a dedicated portal to manage their drug store inventory.
Precondition	The pharmacist must log in to the dedicated pharmacy portal.
Flow of events	1. The system allows the pharmacist to add new medications to the inventory, edit existing medication details, or remove medications from the inventory. 2. The system continuously monitors the quantity of each medication in the inventory. 3. When the quantity of a certain medication falls below a predefined minimum stock level, the system sends a notification to the pharmacist. The notification includes details of the medication and a restocking recommendation. 4. The system provides analytics to the pharmacist, including the total number of drugs sold per day, month, or year. 5. The pharmacist has access to various reports and graphs to analyze their drug store’s performance.

Alternative flow	If there is an error in updating the inventory or generating reports, the system will notify the pharmacist.
Postcondition	The pharmacist has successfully managed their drug store inventory.

Search and filter

Table 4.6 **search and filter use case description**	
Use case name	Medication Search and Filtering
Actors	Customer
Description	User searches for medications and applies filters to refine the search results.
Precondition	The user must access the systems search bar
Flow of events	1. The system provides a user-friendly interface for searching medications. Users can search based on drug name or filter with other relevant drug fields. 2. The system displays medications that match the user's query. 3. Users can refine the search results by applying filters. The available filters are: ○ Users can filter drugs and pharmacies within a specified radius from their current location. ○ Users can filter drugs within a specific price range. ○ Users can filter drugs based on categories, such as over-the-counter or prescription. ○ Users can filter drugs and pharmacies based on user ratings.

Alternative flow	If there is an error during the search or filtering process, the system will notify the user.
Postcondition	The user has successfully searched for medications and applied filters to refine the search results.

Drug order

Table 4.7 **Drug order use case description**	
Use case name	Drug order
Actors	Customer
Description	This use case describes the process of purchasing medication online.
Precondition	The user should log in to the system.
Flow of events	1. The user logs in to the system and selects a medication to view.
	2. The system displays detailed information about the selected medication, including dosage, usage instructions, side effects, and expiration date. 3. The user selects the medication they want to buy. 4. The user selects a delivery option. 5. The user enters their payment information and completes the purchase. 6. The user receives a confirmation with their order details.
Alternative flow	3.1 If the drug requires prescription, the user should not be able to select to buy a drug.
Postcondition	The user has viewed detailed information about the selected medication and/or has paid for ordered medications, including choosing a delivery service.

Registering Pharmacy in Medicine Locator System

Table 4.8 **Registering Pharmacy use case description**	
Use Case name	Registering Pharmacy in Medicine Locator System
Actors	User
Description	This use case outlines the process of a user registering their pharmacy in the Medicine Locator system. The user first registers on the platform, verifies their email address, and then follows a link sent via email to complete the pharmacy registration.
Preconditions	1. The user has access to a valid email address. 2. The Medicine Locator system is accessible.
Flow of events	1. The user navigates to the registration page on the Medicine Locator system.

	2. The user provides necessary information, including their name, contact details, and a password. 3. After successful registration, the system prompts the user to verify their email address. 4. The system sends an email containing a verification link to the user's registered email address. 5. The user checks their email inbox for the verification message. 6. The user clicks on the verification link provided in the email. 7. The Medicine Locator system validates the verification link. 8. If the link is valid, the user's email address is marked as verified in the system. 9. The user navigates to the pharmacy registration section. 10. The user provides information about their pharmacy, including name, address, contact details, and any additional required details. 11. The user submits the pharmacy registration information. 12. The Medicine Locator system displays a summary of the provided information for the user to review. 13. The user confirms the accuracy of the data. 14. The Medicine Locator system sends a confirmation message to the user, indicating the successful registration of their pharmacy.
Alternative flow	If the user encounters issues with the email verification link, they can request a new verification email or contact support for assistance.

Super Admin Registration of Admins

Table 4.9 **Admin registration order use case description**	
Use case name	Super Admin Registration of Admins
Actors	Super Admin

Description	This use case describe how Super admins register admins
precondition	The Super Admin is logged into the system with the appropriate credentials.
Flow of events	1. The Super Admin accesses the system using valid credentials. 2. Once logged in, the Super Admin navigates to the admin registration section. 3. The Super Admin fills in the required information for the new Sub-Admin, 4. The system prompts the Super Admin to review and verify the entered information for accuracy. 5. After confirming the details, the Super Admin submits the registration form to create the new Sub-Admin account. 6. The system validates the provided information, ensuring all required fields are filled and the data is in the correct format. 7. If validation is successful, the system generates unique login credentials (username/password or other authentication methods) for the new Sub-Admin. 8. The system sends a notification to the newly registered Sub-Admin, including login credentials and instructions for accessing the system.
postconditions	The Sub-Admin account is successfully created and accessible in the system. The Super Admin receives a confirmation message indicating the successful registration of the Sub-Admin.

Direction

Table 4.10 **Direction use case description**	
Use case name	Direction Request

Actor	User
Description	This use case describes how a user searches for medications, applies filters to refine the search results, and requests directions to the selected pharmacy.
Preconditions	The user will have to access the system.
Flow of events	1. The system displays medications that match the user's query. 2. Users can refine the search results by applying filters. 3. After selecting a specific medication and pharmacy, the user can request directions to the pharmacy. 4. The system displays the directions to the selected pharmacy on a map.
Alternative flow	If there is an error during the search, filtering, or direction request process, the system will notify the user.
Post condition	The user has successfully searched for medications, applied filters, and received directions to the selected pharmacy.

Reviewing Pharmacy Service

Table 4.11 **Reviewing pharmacy use case description**	
Use case name	Reviewing Pharmacy Service
Actores	Customers,pharmacy
Description	This use case describes the process where a customer reviews and provides feedback on the service received from a pharmacy through the Medicine Locator system. It involves interactions between the customer, the pharmacy, and the Medicine Locator system.

Precondition	The customer has successfully ordered and received drugs from a pharmacy through the Medicine Locator system.
	The pharmacy has fulfilled the customer's order.
Flow of events	1. The customer receives the ordered drugs from the pharmacy. 2. The Drug Locator system sends a notification to the customer, prompting them to provide a review for the recently completed order. 3. The customer provides ratings (e.g., star ratings) for different aspects such as delivery speed, communication, and overall service. 4. The customer writes a detailed review, sharing their experience with the pharmacy's service. They may include comments on the quality of packaging, professionalism of the delivery, and any other relevant aspects. 5. The customer submits the completed review through the Medicine Locator system. 6. The Medicine Locator system notifies the pharmacy about the received review, including the provided ratings and comments. 7. The provided review and ratings are displayed on the pharmacy's profile page within the Medicine Locator system.
Alternative Flow	If the customer chooses not to review the pharmacy, they can skip the review process without affecting the completion of the order.

Post conditions	1. The customer's review is recorded and associated with the specific order. 2. The pharmacy's overall rating and individual reviews are updated in the Medicine Locator system.

Request Refund

Table 4.12 **Request refund use case description**

Usecase name	Requesting Refund for Undelivered Order
Description	In this use case, the scenario where a customer initiates a refund request due to the non-delivery of an order within the pre-specified time frame is outlined.
Actors	customer
Preconditions	1. The customer has successfully placed an order on the Medicine Locator system app. 2. The Medicine Locator has a predefined delivery time commitment for each Pharmacy.
Flow of events	1. The customer, upon realizing that the order has not been delivered within the expected time frame, logs into their account. 2. The customer navigates to the order history section to check the current status of the order. 3. If the order status indicates that it has not been delivered within the pre-specified time, and is able to ask for a refund,the customer proceeds with the refund request.

	4. The customer selects the undelivered order and initiates a refund request through the designated option on the platform. 5. The system validates the customer's refund request by checking if the order falls within the specified criteria for refunds due to non-delivery. 6. If the refund request is valid, the system processes the refund to the customer's original payment method. 7. The customer receives a notification confirming the successful processing of the refund.
Post conditions	The customer receives a refund for the undelivered order.
	The order status is updated to reflect the refund and non-delivery.

4.3.1.3. Use case Scenario

Scenario: User Login Participating Actor: - Mr. john

Flow of events:

1. John, a registered user, opens the online pharmacy app and clicks on the 'Login' button.
2. The system prompts John to enter his username and password.
3. John enters his registered username and password into the respective fields.
4. John clicks on the 'Submit' button.
5. The system validates John's credentials. If the entered username and password match the records in the system's database, John is granted access to his account.
6. If the credentials are incorrect, the system displays an error message and prompts John to try again.

Scenario: User Registration

Participating Actor: - Mr. john

Flow of events:

1. John, a new user, opens the online pharmacy system and clicks on the 'Register' button or wants to buy medication.
2. The system prompts John to enter his personal details such as name, email address, and contact number.
3. John enters his personal details into the respective fields.
4. The system prompts John to create a username and password for his account.
5. John creates a unique username and a strong password, then enters them into the respective fields.
6. John clicks on the 'Submit' button.
7. The system validates the entered information. If the username is unique and the password is strong, the system creates a new account for John.
8. The system sends a confirmation email to John's registered email address.
9. John clicks on the confirmation link in the email to verify his account.
10. The system verifies the account and grants John access to the platform.

Scenario: Purchasing Medication Online

Participating Actor: - Mr. john

Flow of events:

1. John, a registered user, logs into the online pharmacy system using his credentials.
2. The system validates John's credentials and grants him access to the platform.
3. John searches for a specific medication he needs, let's say 'Medication X'.
4. The system displays detailed information about 'Medication X', including its dosage, usage instructions, side effects, and expiration date.
5. John decides to purchase 'Medication X' and push the “Buy” button.
6. The system updates John's cart with 'Medication X'.
7. John reviews his order, sees 'Medication X' listed, and selects delivery options.

8. The system updates the delivery details based on John's selection.
9. John enters his payment information.
10. The system validates John's payment information.
11. After confirming all the details, John proceeds to purchase 'Medication X'.
12. Thesystem processes John's payment, completes the purchase, and deducts 'Medication X' from the inventory.

Scenario : Searching for Medication and Requesting Directions to Pharmacy

Participating Actor : Mr.Abraham(user)

Flow of Events:

1. Abriham opens the system and enters a query for a specific medication.
2. The system processes the query and displays a list of medications that match the Abriham's query.
3. Abriham applies filters to refine the search results.
4. The system updates the list based on the applied filters.
5. Abriham selects a specific medication from the refined list.
6. The system displays a list of pharmacies where the selected medication is available.
7. Abriham selects a specific pharmacy from the list.
8. Abriham requests directions to the selected pharmacy.
9. The system processes the request and displays the directions to the selected pharmacy on a map.

Scenario : Manage Orders

Participating Actor: Mr. Simon(Pharmacist)

Flow of events:

1. Simon logs into the pharmacy system.
2. Simon selects the "Manage orders" option from the system menu.

3. The system displays a list of all drug orders made by users.
4. Simon selects the order that he wants to view.
5. The system displays the order details, including the customer information, order status, medication, and dosage.
6. If Simon Determines that the order details are invalid, he can reject the order by clicking on the "Reject Order" button.
7. The system updates the order status to "Rejected" and sends a notification to the customer about the order rejection.
8. If Simon decides that the order is valid he can accept the order by pushing the “Accept order” button and process the order.

Scenario: Registering Pharmacy in Medicine Locator System

Participating Actor: User (Emily)

Flow of events:

1. Emily visits the registration page on the Medicine Locator system.
2. She provides her name, contact details, and a password, completing the registration process.
3. Upon successful registration, the system prompts Emily to verify her email.
4. An email with a verification link is sent to Emily's registered email address.
5. Emily checks her email inbox and finds the verification message from the Medicine Locator system.
6. Emily clicks on the verification link provided in the email.
7. The Medicine Locator system validates the verification link.
8. The system marks Emily's email address as verified.
9. She navigates to the pharmacy registration section.
10. Emily provides accurate information about her pharmacy, including name, address, and contact details.
11. Emily submits the pharmacy registration information.
12. The system displays a summary of the provided information for Emily to review.

13. She confirms the accuracy of the data.
14. Emily receives a confirmation message indicating the successful registration of her pharmacy in the Medicine Locator system.

Scenario : Pharmacy Registration Review

Participating Actor: Mr. Amanuel (Admin)

Flow of events:

1. Amanuel logs into the system.
2. The system displays a list of pending pharmacy registration requests.
3. Amanuel selects a pharmacy registration request to review.
4. The system displays the pharmacy's information, including the pharmacy's name, address, and license.
5. Amanuel reviews the pharmacy's information and verifies that the pharmacy is licensed and in good standing.
6. Amanuel decides to approve or reject the pharmacy registration request.
7. If Amanuel approves the pharmacy registration request, the system grants the pharmacy access to the system's features.
8. If Amanuel rejects the pharmacy registration request, the system denies the pharmacy access to the system's features.
9. The system sends an email notification to the pharmacy informing them of the decision.

Scenario Title: Super Admin Registration of Admins

Participating Actor: Sarah(Super Admin)

Flow of Events:

1. Sarah logs into the project management system using her Super Admin credentials.
2. Once inside the system, Sarah navigates to the "Admin Management" section to initiate the registration process for a new Sub-Admin.

3. Sarah enters the required information for Alex, the new Sub-Admin:
4. The system prompts Sarah to review the entered details for accuracy.
5. Sarah confirms that all information is correct and moves on to the next step.
6. Sarah submits the registration form, indicating her approval for creating the Sub-Admin account for Alex.
7. The system validates the provided information, ensuring all fields are filled correctly and in the required format.
8. Upon successful validation, the system generates unique login credentials for Alex, such as a username and password.
9. The system sends an automated email to Alex, welcoming him to the system and providing the login credentials.
10. The email also includes instructions on how to access the project management system.
11. Sarah receives a system-generated confirmation message indicating that the registration process for Alex was successful.

Scenario Title: Providing Feedback on Pharmacy Service

Participating Actor: Mr. Bob (customer),pharmacy

Flow of Events:

1. Bob receives the prescribed drugs from the pharmacy.
2. The Medicine Locator system sends a prompt to Bob, asking him to review the recently completed order.
3. Bob provides ratings for various aspects of the pharmacy's service, including delivery speed, communication, and overall satisfaction.
4. Bob writes a detailed review, expressing his experience with the pharmacy. He appreciates the quick delivery and the professionalism of the pharmacy staff.
5. Bob submits the review through the Drug Locator system.

6. The Medicine Locator system notifies the pharmacy about the received review, including the provided ratings and comments.
7. Bob's review and ratings are displayed on the pharmacy's profile page within the Medicine Locator system.

Scenario:Request Refund

Participating Actor: Mr. john (customer)

Flow of events:

1. John has an account on the Medicine Locator system.
2. John places an order for a medicine on the system with an expected delivery time of 2 business days.
3. John receives an order confirmation email with details and the expected delivery date.
4. On the second business day, John realizes that the medicine has not been delivered as per the promised time frame.
5. John logs into his account.
6. Navigating to the order history section, John checks the status of the medicine order.
7. John selects the undelivered medicine order and initiates a refund request through the platform's refund request option.
8. The medicine locator system validates John's refund request by confirming that the medicine has indeed not been delivered within the specified time.
9. Since the request is valid, the system processes the refund to John's original payment method.
10. John receives a notification on the platform confirming the successful processing of the refund.

4.4. Object Model

An object model is a description of an object-oriented architecture, including the details of the object structure, interfaces between objects, and other object-oriented features and functions. In this section, we discuss the object model class diagram and data dictionary. In the class diagram, we have described diagrammatically the conceptual relationship among objects and class of the proposed system Drug Locator system . Metadata that describes the database of the Drug Locator system that includes data ownership, data relationships to other objects, and other data; thus, metadata are discussed under data dictionaries.

4.4.1. Class Diagram

The conceptual class diagram serves as the static representation of the Medicine Locator System, providing a visual overview, documentation, and executable code generation. It outlines attributes, operations, and constraints of each class, making it a key tool for modeling object-oriented systems. In the context of the Medicine Locator System, this diagram is foundational for development, offering a concise depiction of static elements and seamless integration with object-oriented programming practices.

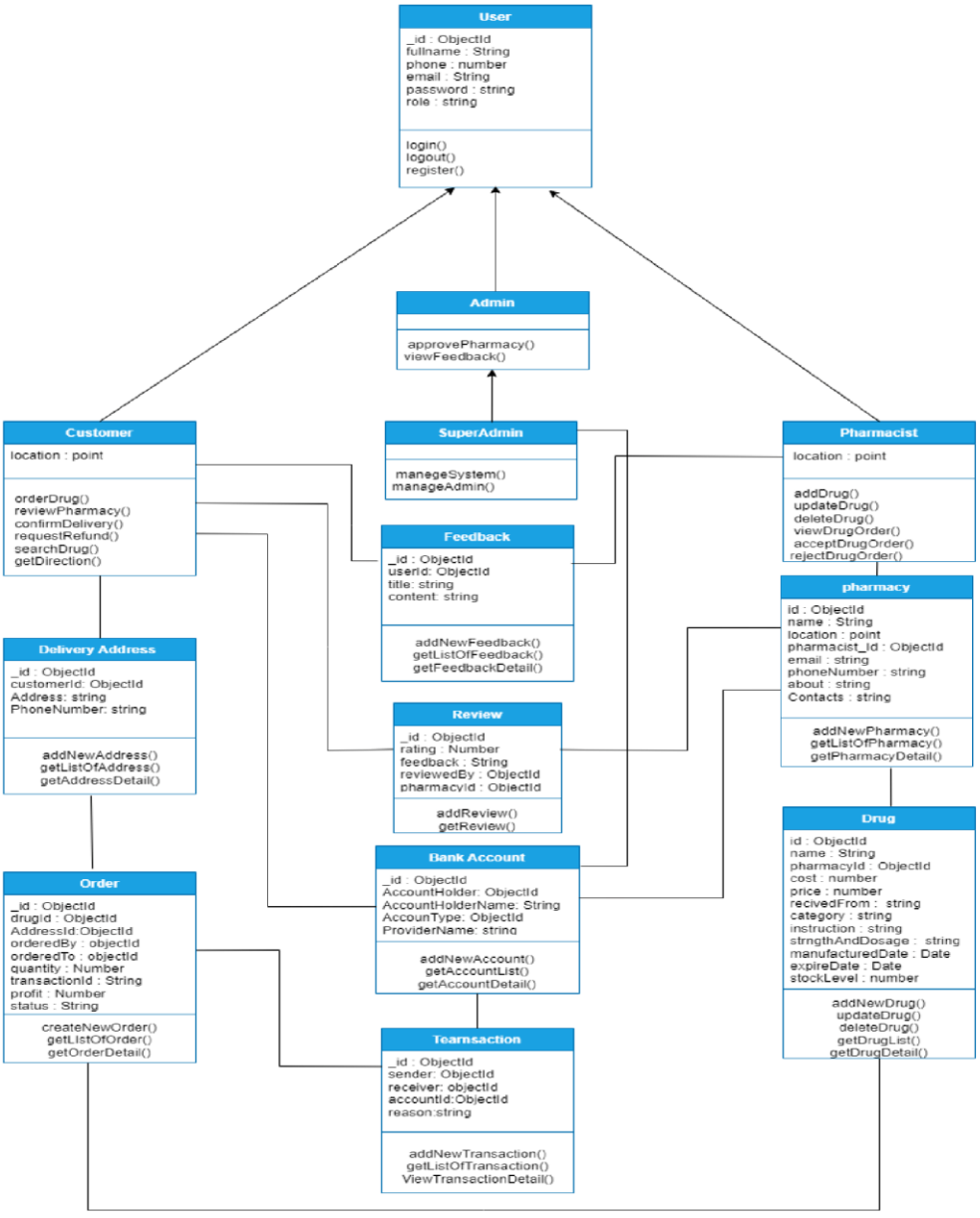


Fig 4.0.2 conceptual class diagram

4.5. Data Dictionary

User collection

Table 4. 1 user collection data dictionary

Field	Data Type	Format	Key	Constraint
			Constraint	
_id	ObjectId	-	Pk	Not null
FullName	String	-	-	Not null
email	String	unique	-	Not null
phoneNumber	String	unique	-	-
password	String	-	-	Not null
location	Point	Coordinate:[longitude, latitude]	-	Not null
role	string	enum:[‘customer’, ‘Pharmacist’, ‘admin’]	-	Not null

Pharmacy collection

Table 4. 2 pharmacy collection data dictionary

Field	Data Type	Format	Key Constraint	Constraint
_id	ObjectId	-	Pk	Not null
pharmacistId	ObjectId		Fk refer user	
name	String	-	-	Not null
email	String	unique	-	Not null

phoneNumber	String	-	-	-
location	Point	Coordinate:[longitude, latitude]	-	Not null
about	string		-	Not null

Drug collection

Table 4. 3 drug collection data dictionary

Field	Data Type	Format	Key Constraint	Constraint
_id	ObjectId	-	Pk	Not null
pharmacyId	ObjectId	-	Fk refer pharmacy	Not null
name	String	-	-	Not null
recivedFrom	String			Not null
strengthAndDosage	string			Not null
ingredient	Array	[‘string’]	-	Not null
category	String	-	-	Not null
Side Effect	String	-	-	Not null
instruction	String	-	-	Not null
munufacturedDate	Date	-	-	Not null

Expire date	Date	-	-	Not null
selling_price	Number	-	-	Not null
cost	Number	-	-	Not null
Stock level	Number	-	-	Not null
needPrescription	Boolean	-	-	Not Null
minStock level	Number	-	-	Not null

Drug_order collection

Table 4. 4 drug order collection data dictionary

Field	Data Type	Format	Key Constraint	Constraint
_id	ObjectId	-	Pk	Not null
drugId	ObjectId	-	Pk refer Drug	Not null
orderedby	ObjectId	-	Fk refer User	Not null
oderedto	ObjectId		Fk refer Pharmacy	Not null
quantity	Number	-	-	Not null
paymentMethod	String			Not null
transactionId	ObjectId	-	-	Not null
profit	Number	-	-	-

Status	String	Enum:[in-progress,delivered, aborted]	-	Not null
deliveredDate	Date			Not null

Review collection

Table 4. 5 review collection data dictionary

Field	Data Type	Format	Key Constraint	Constraint
_id	ObjectId	-	Pk	Not null
reviewedBy	ObjectId	-	Pk refer User	Not null
pharmacyId	ObjectId		Fk refer Pharmacy	Not null
rating	Number	-	-	Not null
feedback	String	-	-	-

Delivery Address

Table 4.8 delivery Address collection data dictionary

Field	Data Type	Format	Key Constraint	Constraint
_id	ObjectId	-	Pk	Not null
customerId	ObjectId	-	Pk refer Customer	Not null
phoneNumber	Number		-	Not null
Address	String	-	-	Not null

Table 4. 6 Feedback Address collection data dictionary

Feedback/

Field	Data Type	Format	Key Constraint	Constraint
_id	ObjectId	-	Pk	Not null

reviewedBy	ObjectId	-	Pk refer User	Not null
pharmacyId	ObjectId		Fk refer Pharmacy	Not null
rating	Number	-	-	Not null
feedback	String	-	-	-

Table 4. 7 Account collection data dictionary

Account

Field	Data Type	Format	Key Constraint	Constraint
_id	ObjectId	-	Pk	Not null
accountHolder	ObjectId	-	Pk refer or User pharmacy	Not null
accountName	string		-	Not null
accountType	string	-	-	Not null
providerName	string	-	-	Not null

Table 4. 8 Transaction collection data dictionary

Transaction

Field	Data Type	Format	Key Constraint	Constraint
_id	ObjectId	-	Pk	Not null
sender	String	-	-	Not null
Receiver	string		-	Not null

accountId	ObjectId	-	Fk refer Account	Not null
reason	String	-	-	-

4.5 Dynamic Modeling

The dynamic model illustrates the behavior and collaboration among objects in the Medicine Locator System. Focusing on two key dynamic model diagrams, namely sequence and activity diagrams, these provide insights into the operational flow and interactions. The sequence diagram portrays the chronological message exchange between objects, while the activity diagram visually represents the workflow during different scenarios. Together, these dynamic model diagrams enhance our understanding of the Medicine Locator System's functionality, complementing the conceptual class diagram and facilitating a comprehensive development approach.

4.5.1. Sequence diagram

A sequence diagram highlights the temporal ordering of messages, depicting objects, classes, and the sequential message exchange in the Medicine Locator System. This visual representation clarifies the interaction dynamics and showcases how elements collaborate to execute scenario functionality in the Drug Locator system.

Pharmacy registration sequence diagram

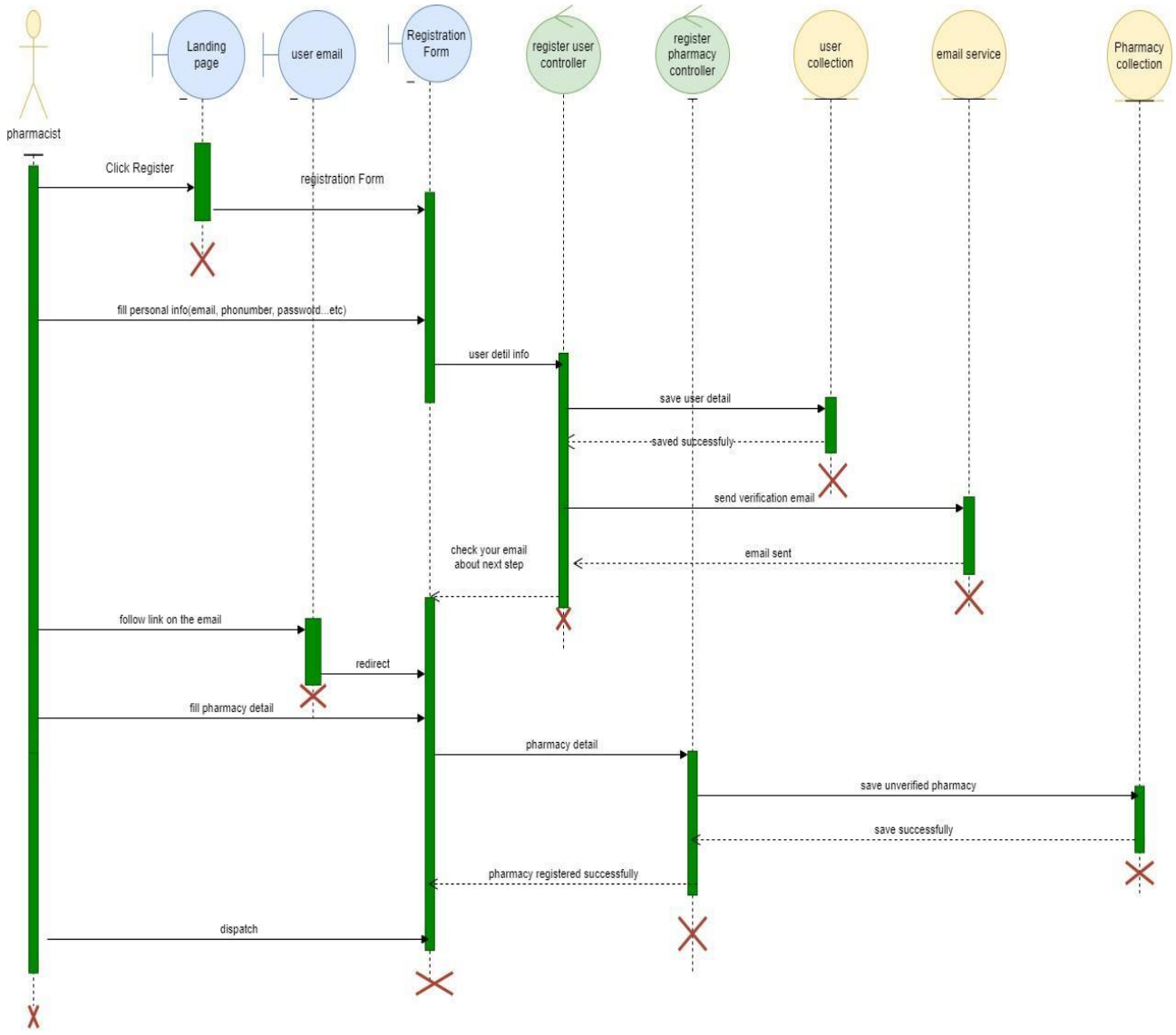


Fig 4.3 Pharmacy registration sequence diagram

Pharmacy recommendation sequence diagram

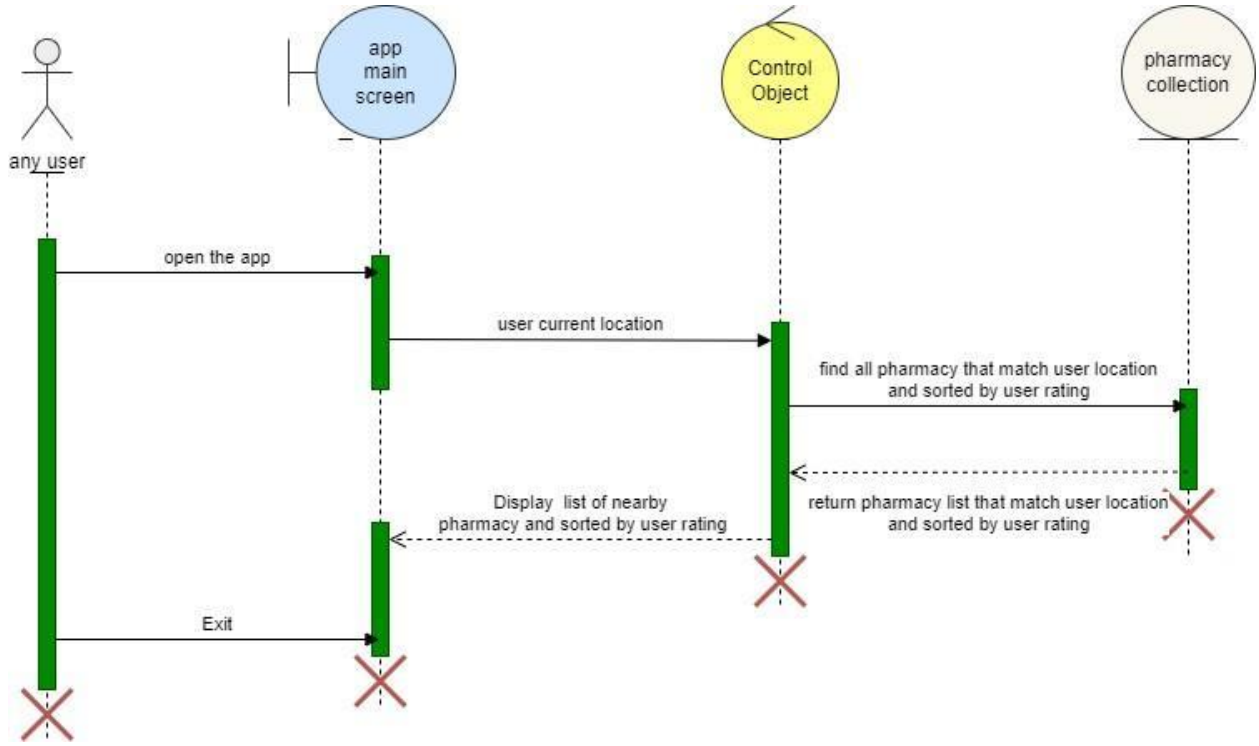


Fig 4.4 Pharmacy recommendation sequence diagram

Admin pharmacy approval sequence diagram

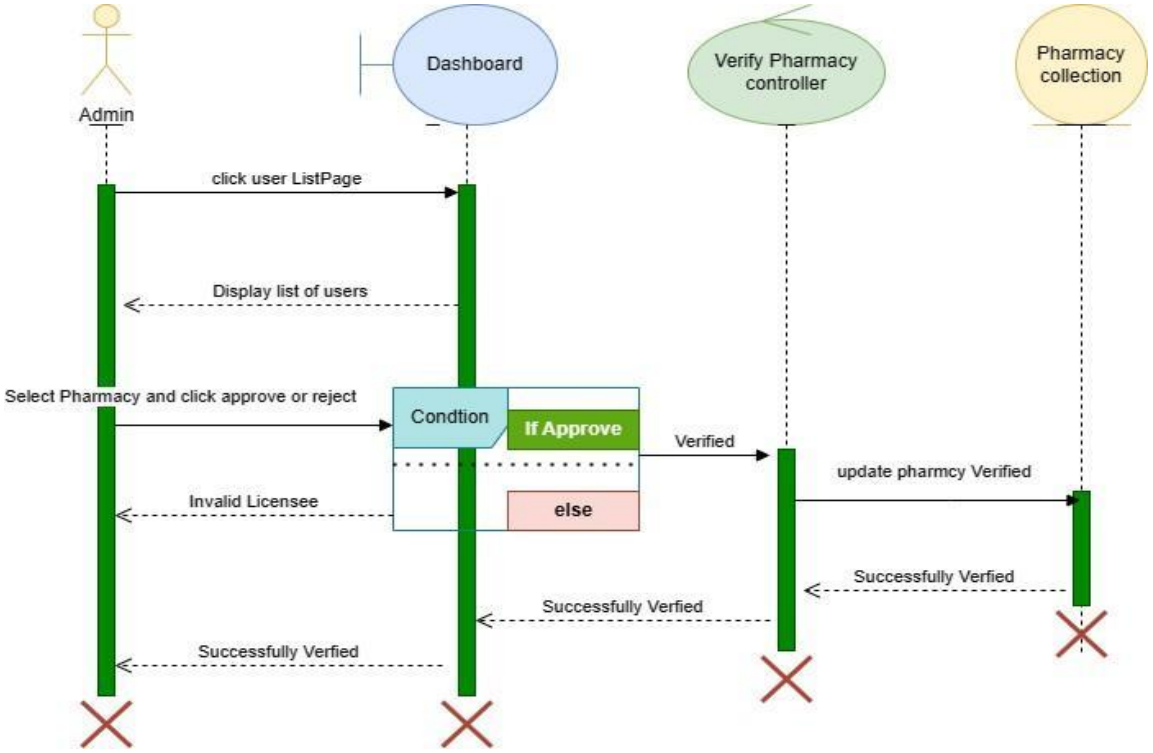


Fig 4.5 Admin pharmacy approval sequence diagram

Drug search and filter sequence diagram

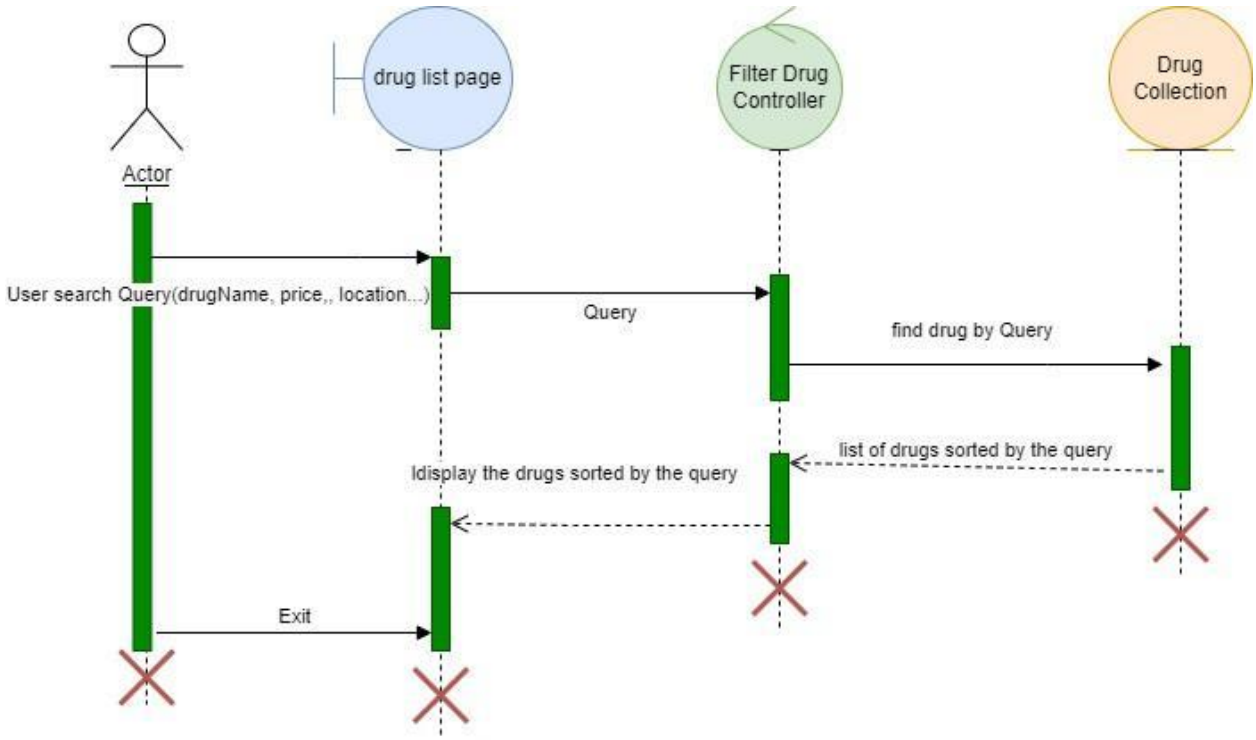


Fig 4.6 Drug search and filter sequence diagram

Drug order sequence diagram

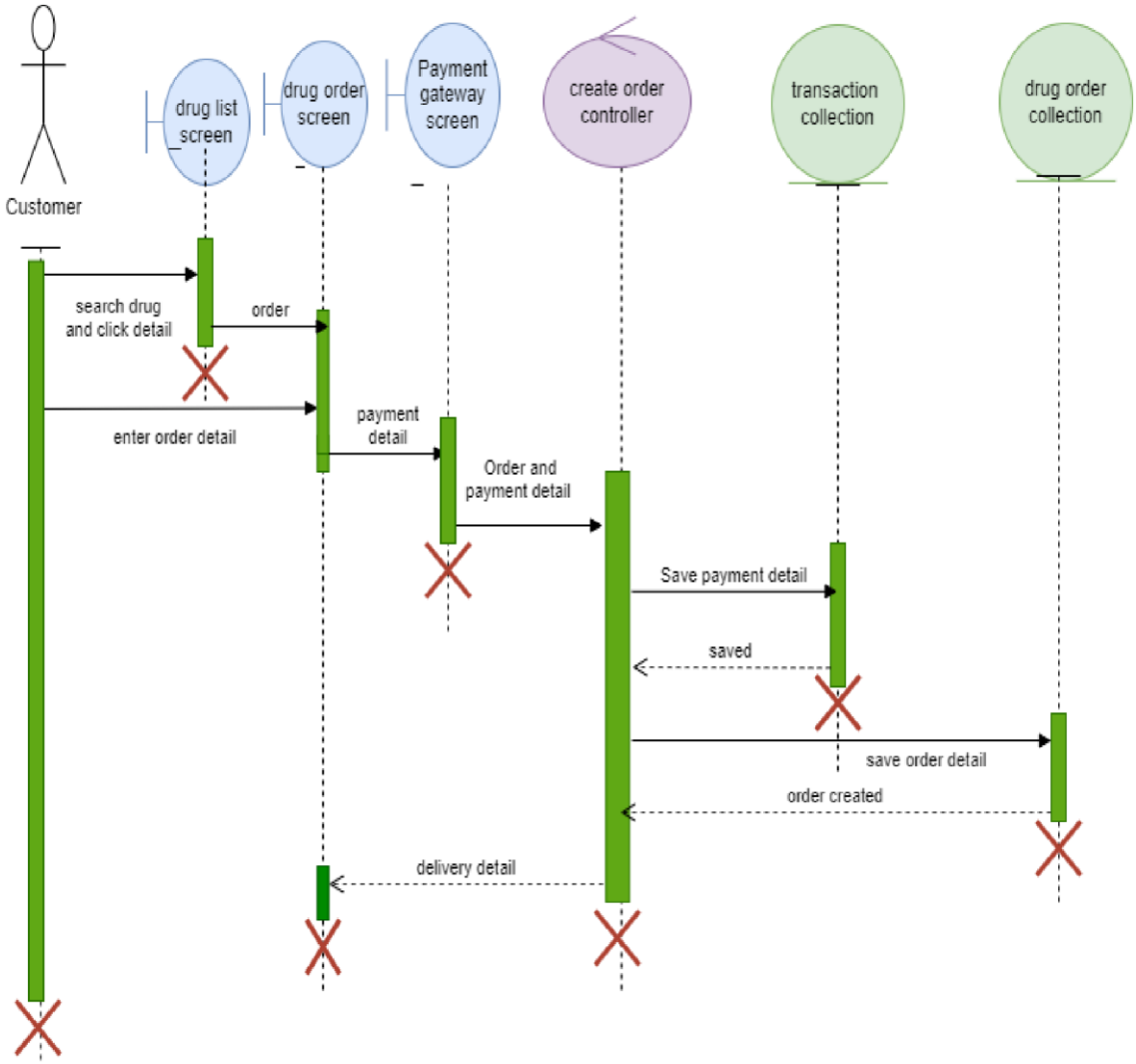


Fig 4.7 Drug order sequence diagram

Conform delivery sequence diagram

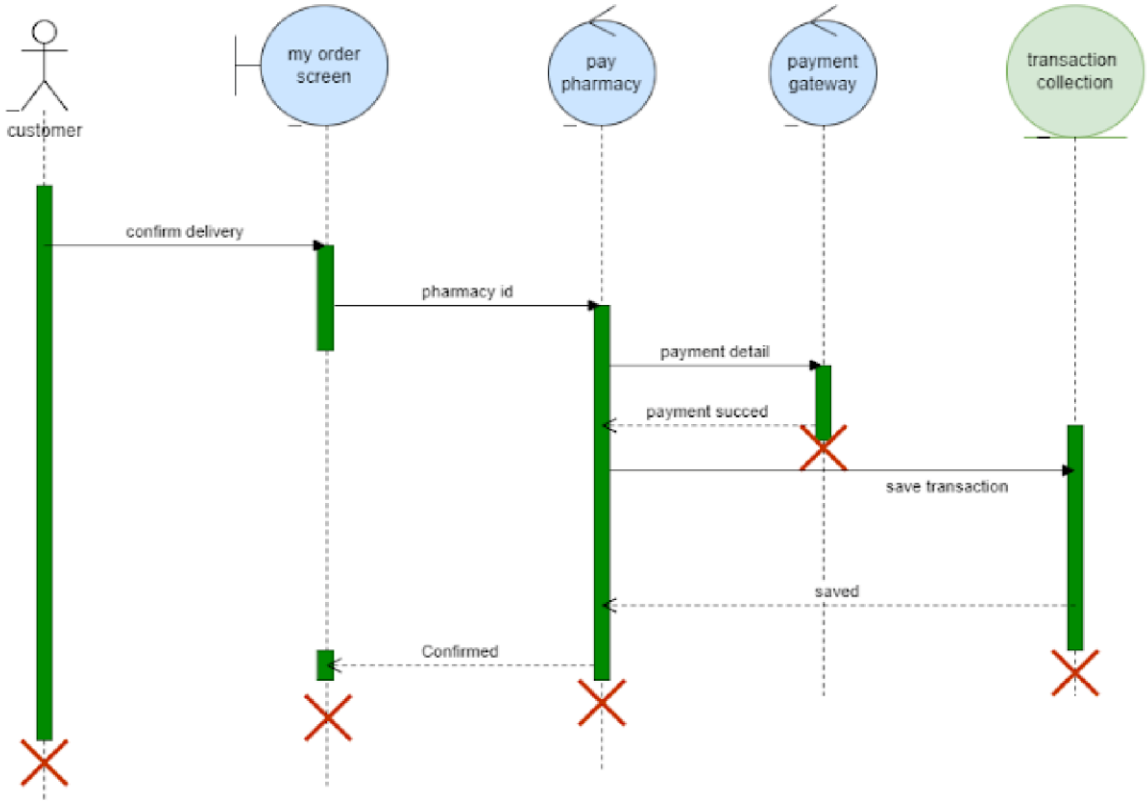


Fig 4.8 conform delivery sequence diagram

Cancel Order if Delivery Date has Expired - Sequence Diagram

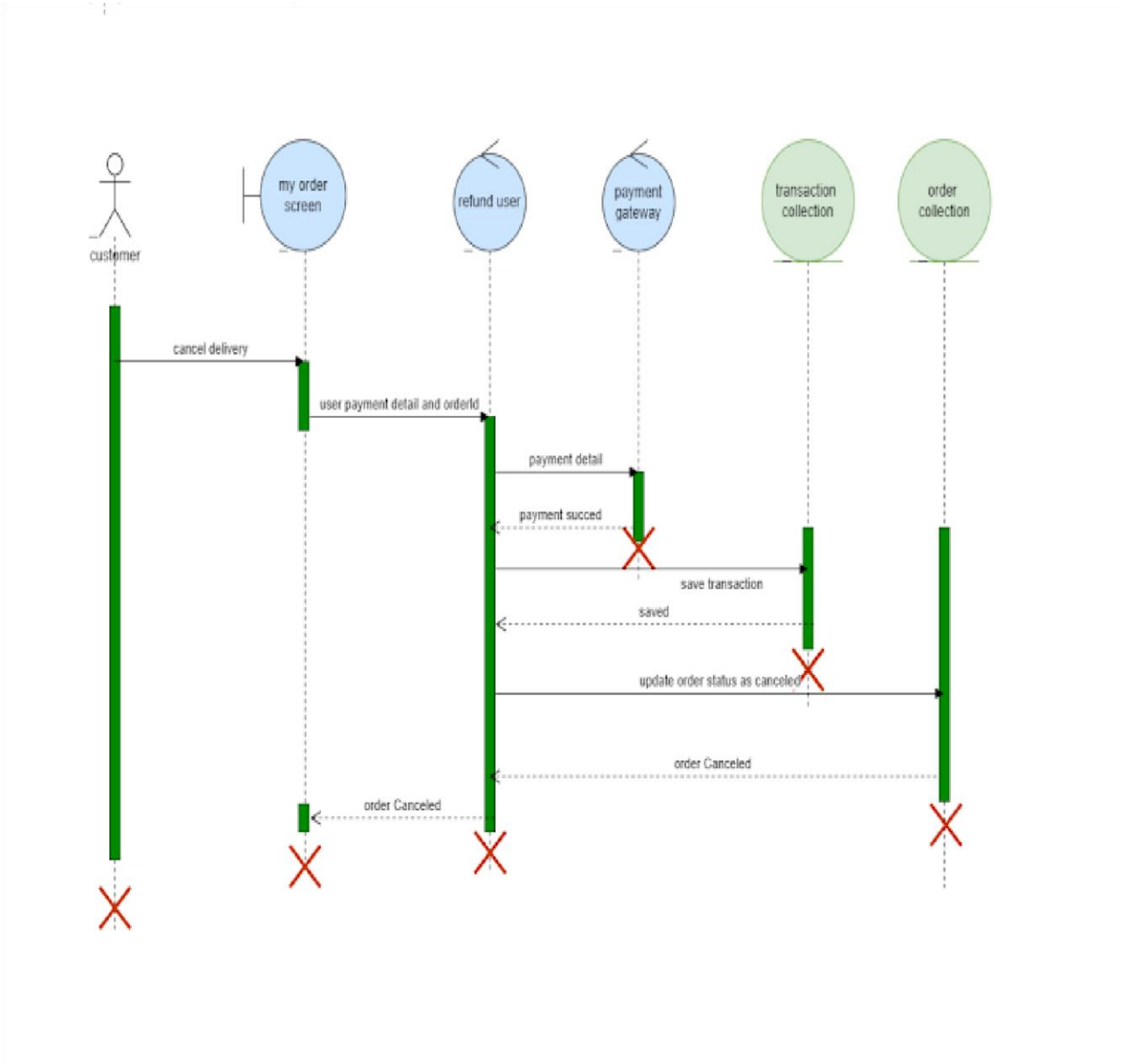
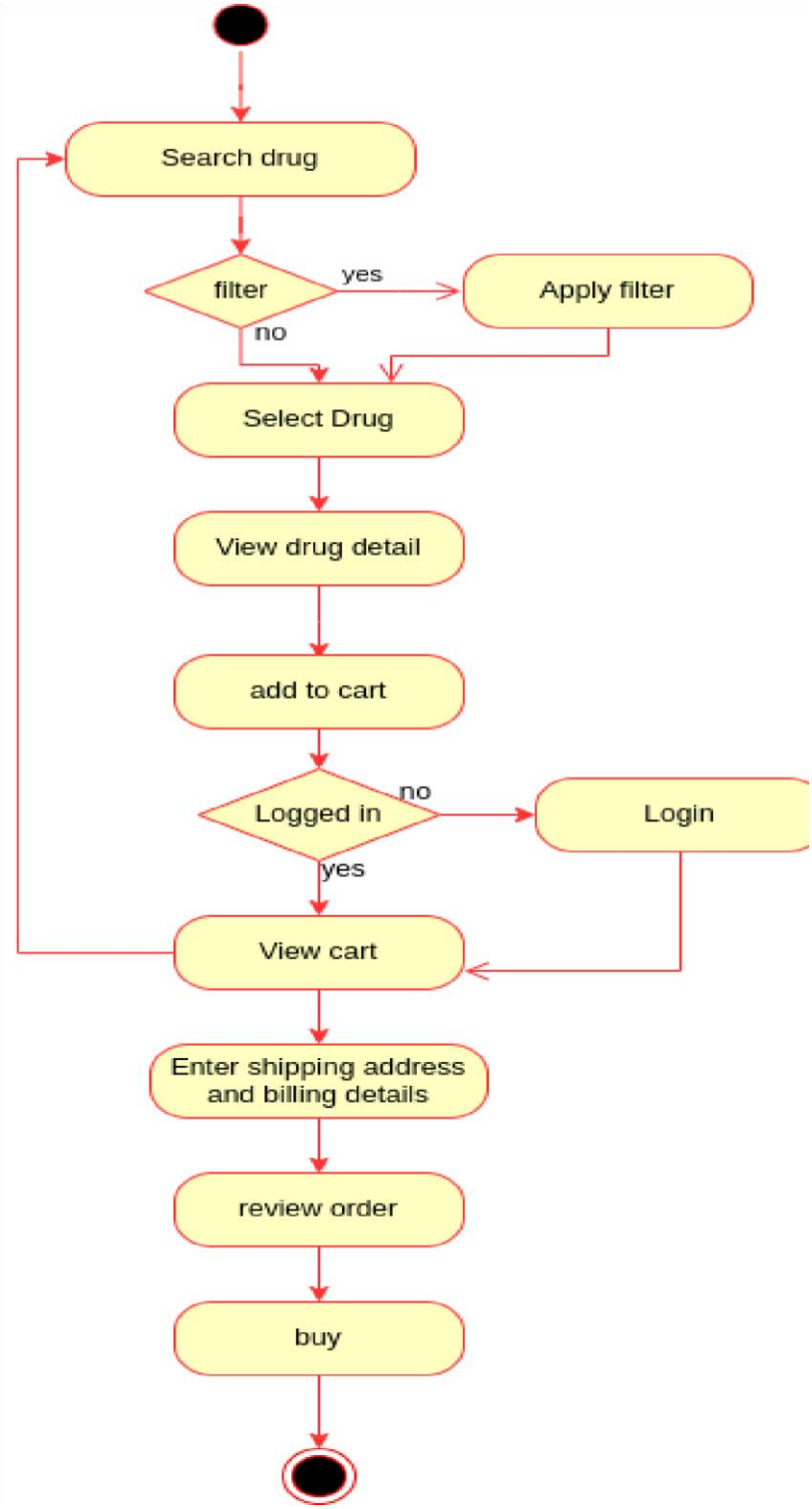


Fig 4.9 Cancel Order if Delivery Date has Expired - Sequence Diagram

4.5.2. Activity Diagram

An activity diagram serves as a flow chart, illustrating the progression from one activity to another within the Medicine Locator System. We specifically detail the drug search and Order activity diagrams to demonstrate the system's operations in executing tasks assigned to actors in the Drug Locator system.

Drug search and Order



4.10 Drug search and Order activity diagram

Manege orders

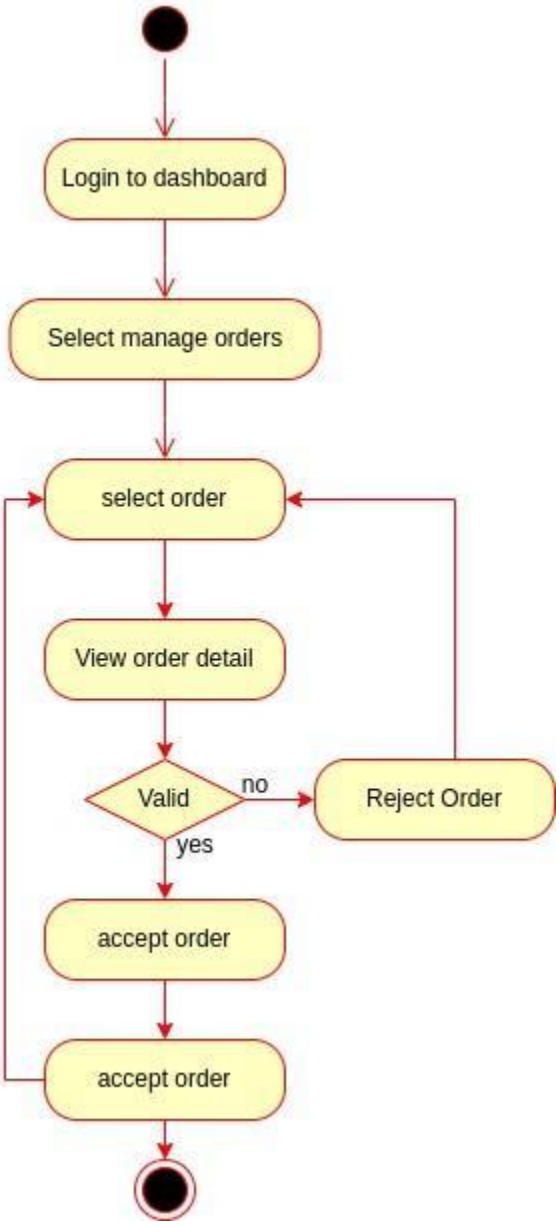
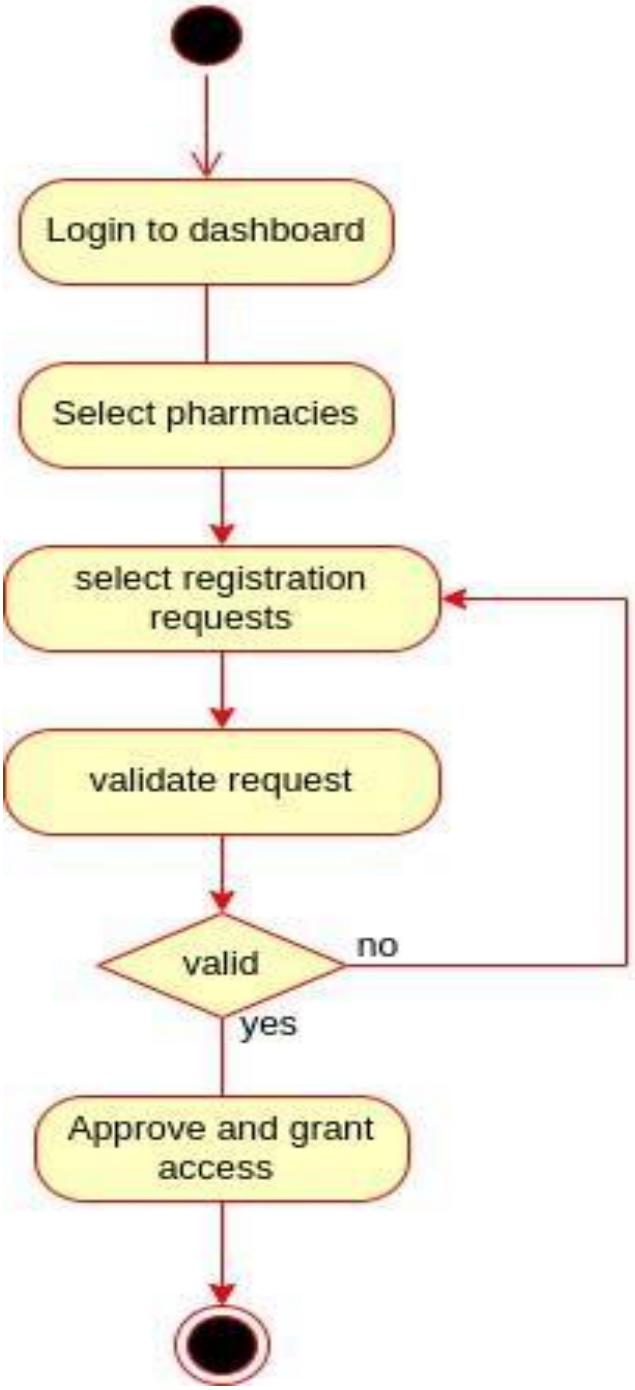


Fig 4.11 pharmacy manage orders activity diagram

Admin Validate Pharmacies



4.12 Admin validate pharmacies activity diagram

add Drugs

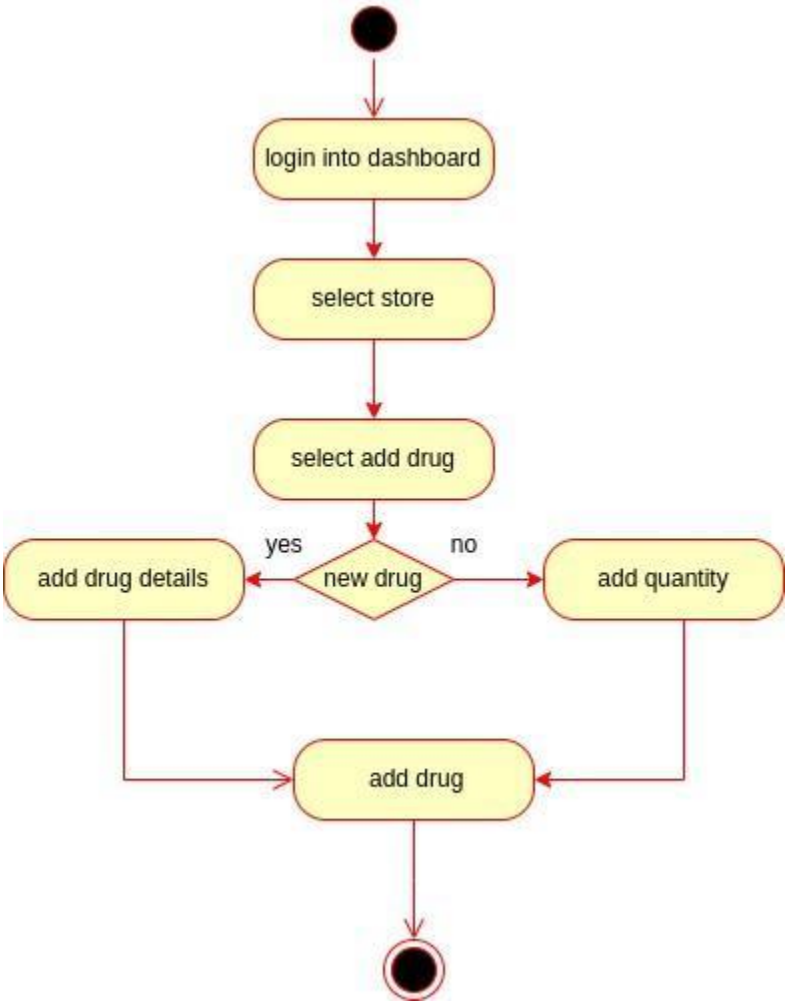


Fig 4.13 pharmacy add drugs activity diagram

4.5.3. State Chart Diagram

Drug Order state chart diagram

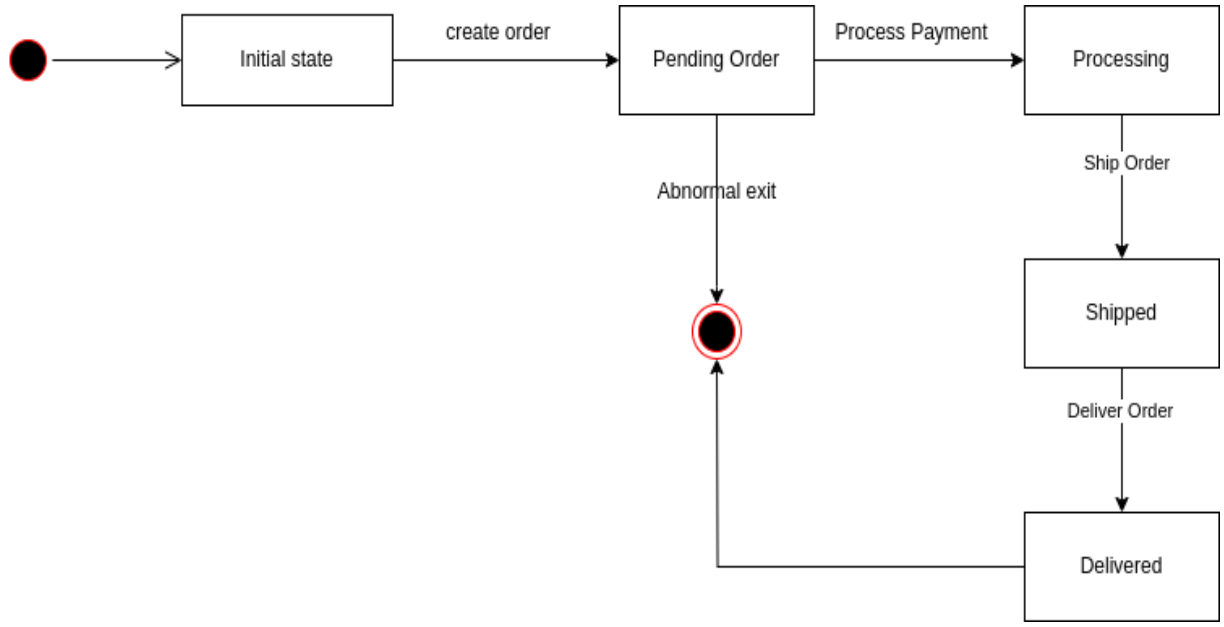
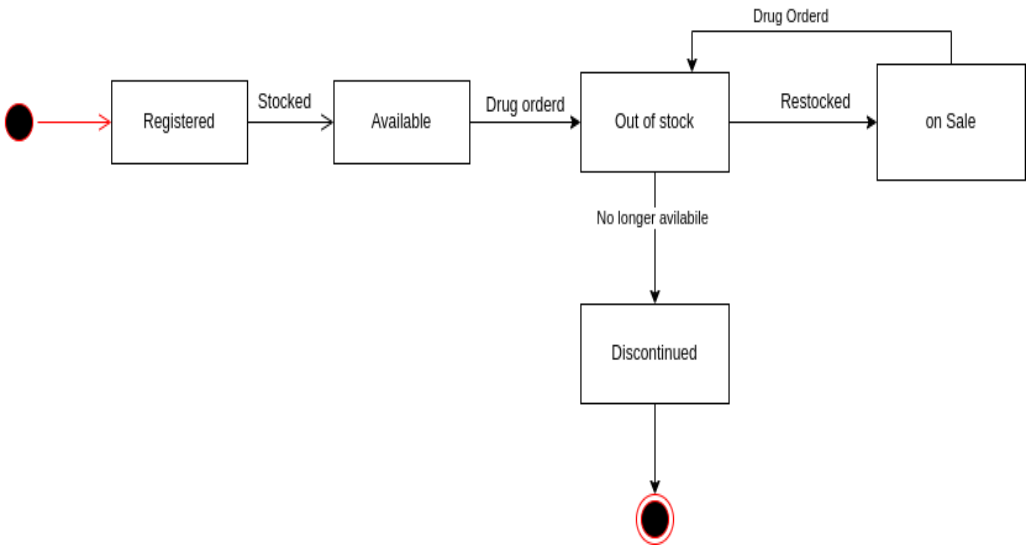


Fig 4.14 Drug order state chart diagram

Drug state chart diagram



4.15 Drug state chart diagram

CHAPTER FIVE

SYSTEM DESIGN

The Systems design of this project defines the elements of the Drug Locator system like modules, architecture, components, and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing, and designing systems that satisfy the specific needs and requirements of the stakeholder and also defines the coherent and well running system.

5.1. Design Goals

The design goals are derived from the non-functional requirements of the system, which describes the Drug Locator system that specifies the qualities of the system that should be achieved and addressed during the design of the system.

Based on the systems stakeholder and non-functional requirement the Medicine Locator System design goal are listed below:

User Interface and Human Factors:- Design an intuitive and user-friendly interface.

Hardware Consideration:- Optimize the system to work efficiently on a variety of devices.

Security Issues:- Implement secure authentication and authorization mechanisms.

Performance Consideration:- Achieve fast response times for locating medicines.

Error Handling and Validation:- Provide meaningful error messages for users and implement thorough input validation to prevent data inconsistencies.

Quality Issues:- Ensure code quality through consistent coding standards.

Backup and Recovery:- Implement a robust backup mechanism for system data.

Physical Environment:- Design the system to function in different network conditions.

Resource Issues:- Optimize resource usage to minimize server costs.

Documentation:- Maintain comprehensive documentation for system components.

These design goals aim to address various aspects of the Medicine Locator System, ensuring a well-rounded and robust system that meets user needs while considering technical, security, and environmental factors.

5.2. Proposed System Architecture

Medicine Locator System is underpinned by a well-defined system architecture that ensures a cohesive and efficient deployment of functionality across various subsystems. Embracing a 3-tier Client/Server Architecture, the system is organized into three distinct layers, each with its designated set of responsibilities:

1. Presentation Tier:

Functionality Assignment:

- ➔ User Interface (UI) development for both web and mobile applications.
- ➔ Rendering and presenting information to end-users.
- ➔ Capturing and processing user inputs.

Rationale:

- ➔ This tier is specifically designed to handle the user interaction aspect of the system. By assigning UI-related tasks to the presentation tier, we streamline the development of a responsive and intuitive interface for seamless user engagement.

2. Business Logic Tier:

Functionality Assignment:

- ➔ Implementation of business rules and logic.
- ➔ Coordination of tasks and communication between presentation and data tiers.
- ➔ Workflow management and processing of critical system functionalities.

Rationale:

- ➔ The business logic tier serves as the brain of our Medicine Locator System. By assigning critical business-related tasks to this layer, we ensure a centralized and organized handling of complex operations, promoting clarity and maintainability.

3. Data Storage Tier:

Functionality Assignment:

- ➔ Management and storage of medical data.
- ➔ Interaction with the database management system for data operations.
- ➔ Ensuring data integrity and security measures.

Rationale:

- ➔ Safeguarding sensitive medical information is paramount. The data storage tier focuses on efficient and secure handling of data, ensuring that the system's information remains accurate, available, and protected against unauthorized access.

Why 3-Tier Architecture for the Medicine Locator System?

Our adoption of a 3-tier architecture for the Medicine Locator System is strategically chosen for the following advantages:

- ➔ **Scalability**: The structured 3-tier approach seamlessly scales individual layers, catering to the evolving needs of web and mobile applications.
- ➔ **Maintainability**: A modular design enables independent updates to each tier, minimizing risks, and facilitating efficient debugging and maintenance.
- ➔ **Flexibility**: Separation of concerns allows for adaptability to changes in user interfaces, business rules, and data structures without disruptions.
- ➔ **Security**: Isolating business logic and data storage layers adds an extra security layer, ensuring robust protection against potential threats.

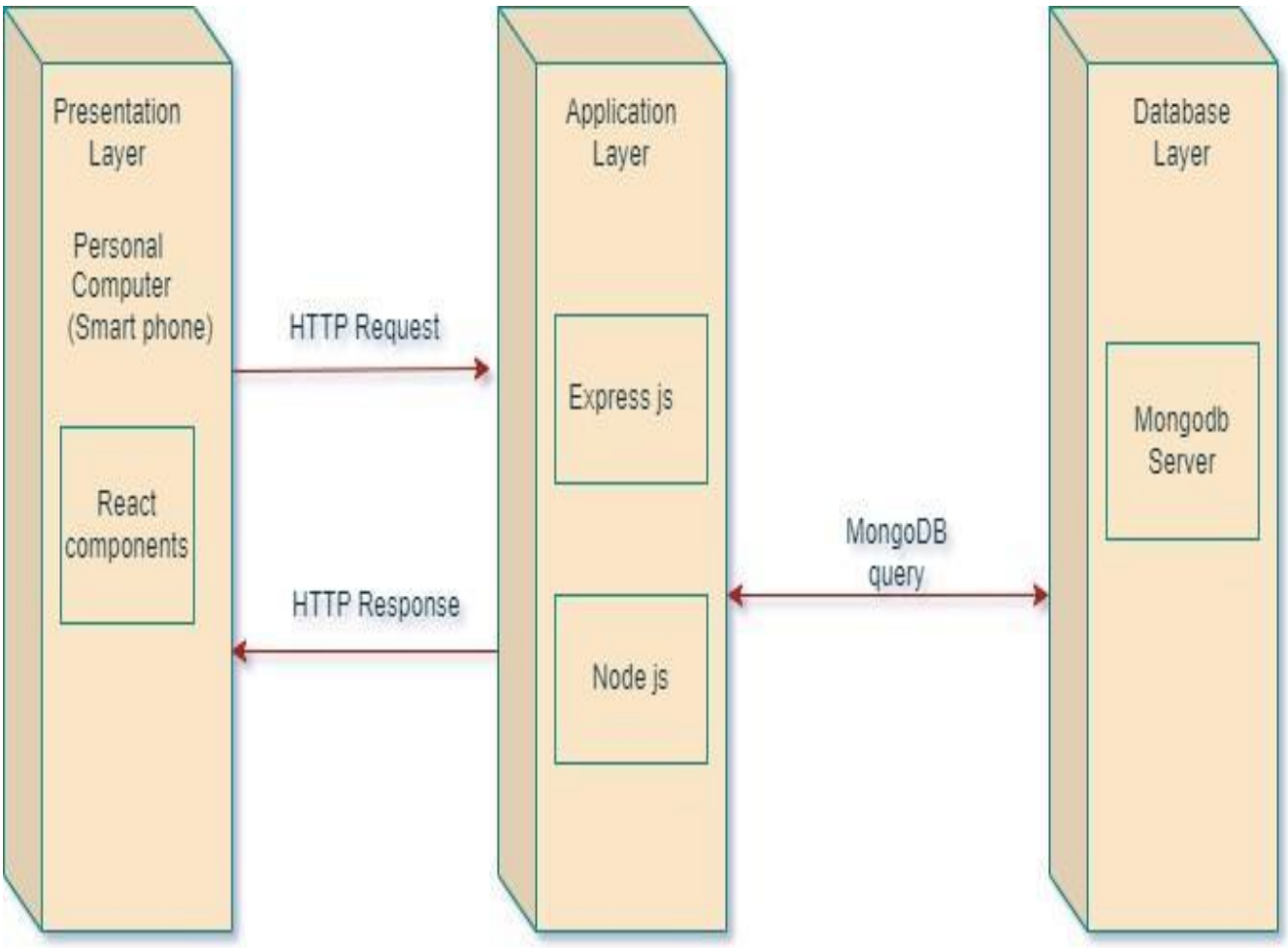


Fig 5.1 System Architecture Diagram

5.2.1. Subsystem Decomposition and Description

A Medicine Locator System typically consists of several subsystems that work together to provide a comprehensive solution for users. Below is a decomposition of subsystems with descriptions for a Medicine Locator System:-

1. User management :- This is a subsystem for managing users and it includes other components such as user registration, authentication, user updation and others.
2. Search and Filter :- It is a subsystem for searching and filtering drugs and it includes other components such as search, filter, locate and others.

3. Order Management:- This is a subsystem for managing orders and it includes other components such as payment, report, refund, delivery and others.
4. Inventory Management:-This is a subsystem for managing Inventory and it includes other components such as adding drugs, , updating drugs, View details and others.
5. Pharmacy Management:- It is a subsystem for managing pharmacy and it includes other components such as pharmacy approval and rejection.
6. Feedback Management:- It is a subsystem for managing Feedbacks and it includes other components such as add, view and resolve feedbacks.

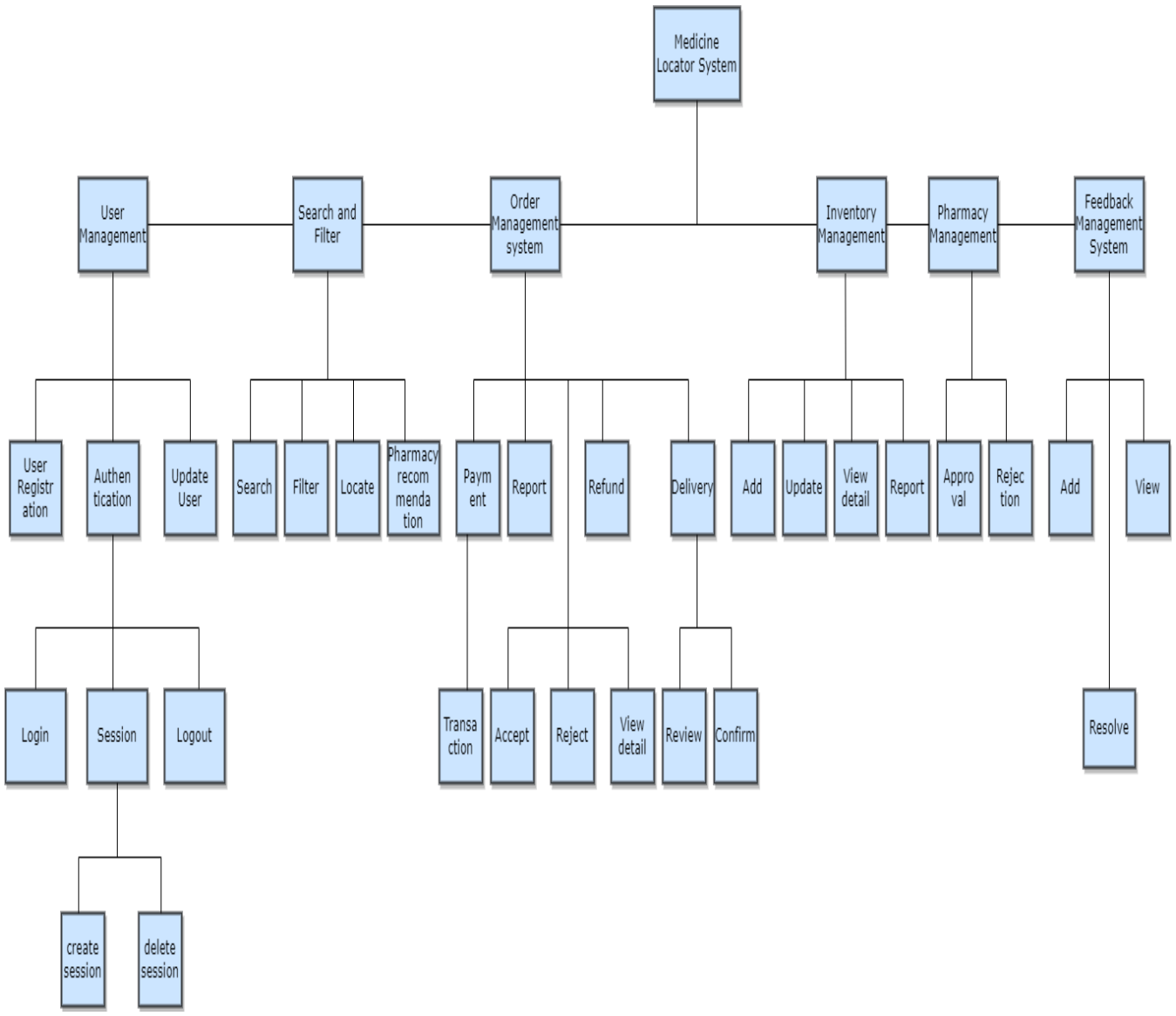


Fig 5.2 system decomposition diagram

5.2.2. Hardware/Software Mapping

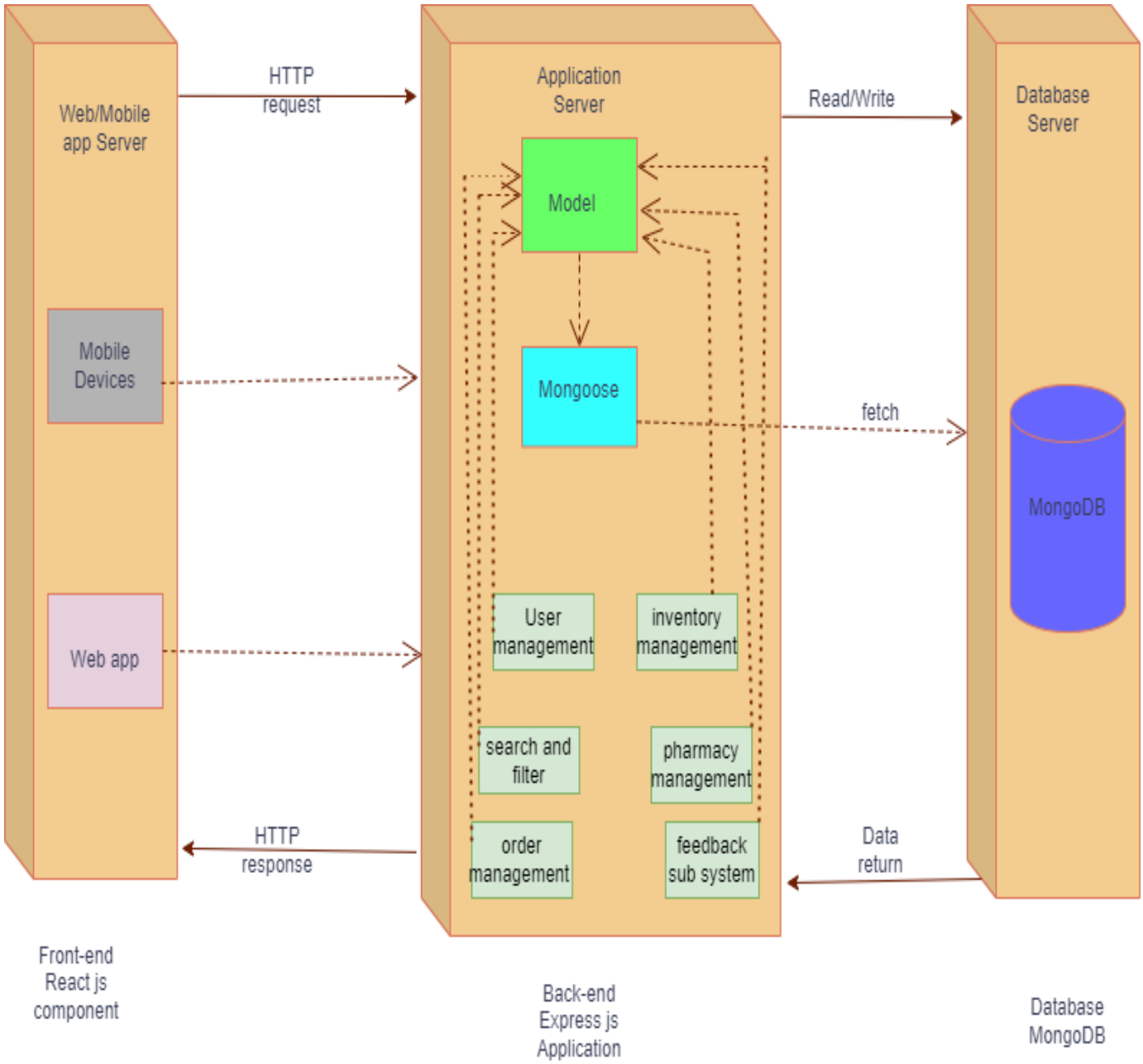


Fig 5.3 Deployment diagram

5.3.4 Detailed Class Diagram

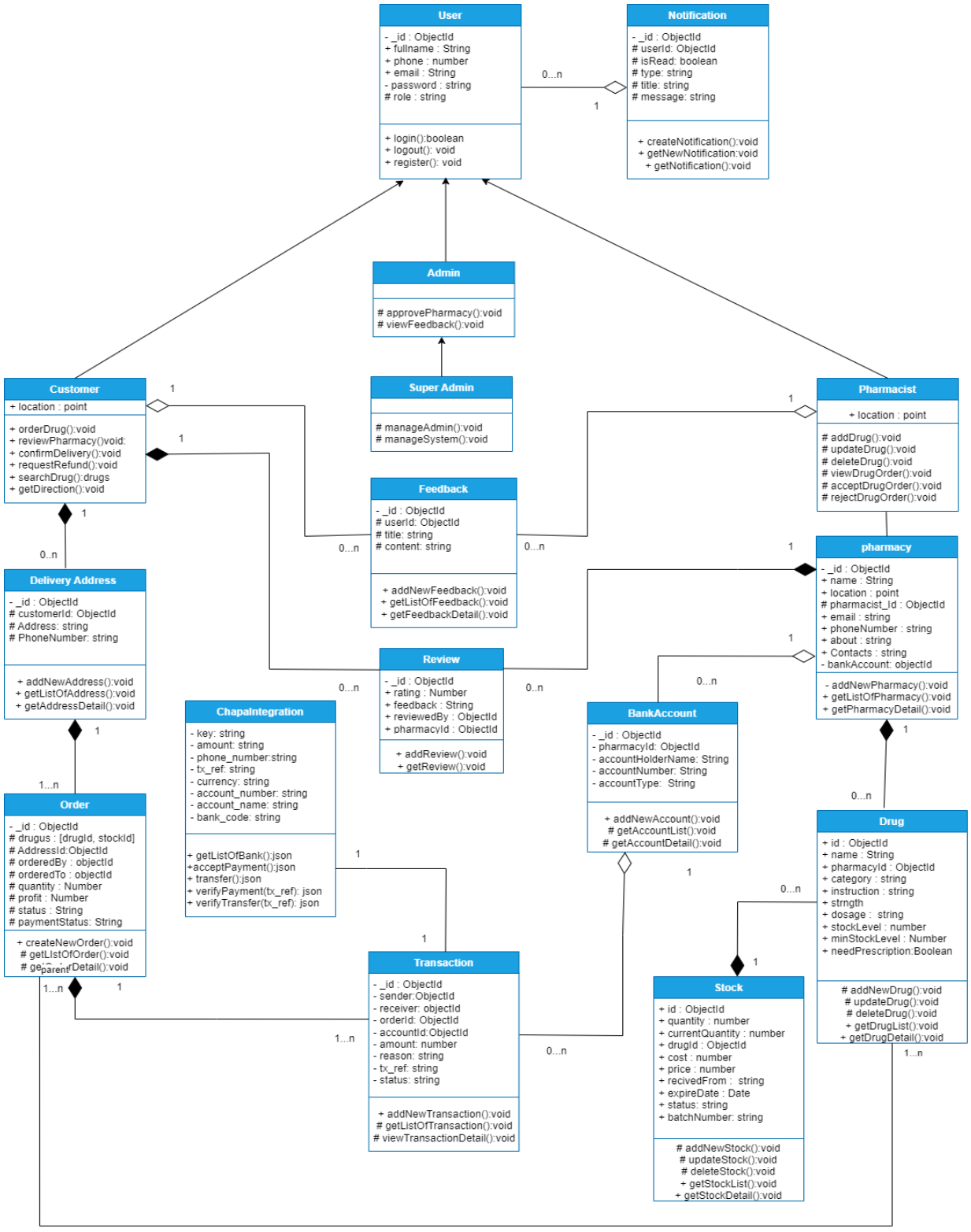


Fig 5.4 detail class diagram

5.3.5. Persistent Data Management

Medicine Locator, a robust system designed for precise pharmaceutical information retrieval, heavily relies on a NoSQL database, specifically MongoDB. As a non-tabular database, MongoDB brings advantages such as flexible schemas, horizontal scaling capabilities, rapid query processing, and a developer-friendly environment.

Within the Medicine Locator system, MongoDB serves as the central repository for all metadata associated with client-server interactions..

To fortify data security, a daily backup protocol is implemented, safeguarding the integrity and availability of crucial information within the Medicine Locator system. This precautionary measure is essential for maintaining the reliability of the system, mitigating potential data loss scenarios, and ensuring the uninterrupted functionality of the Medicine Locator application.

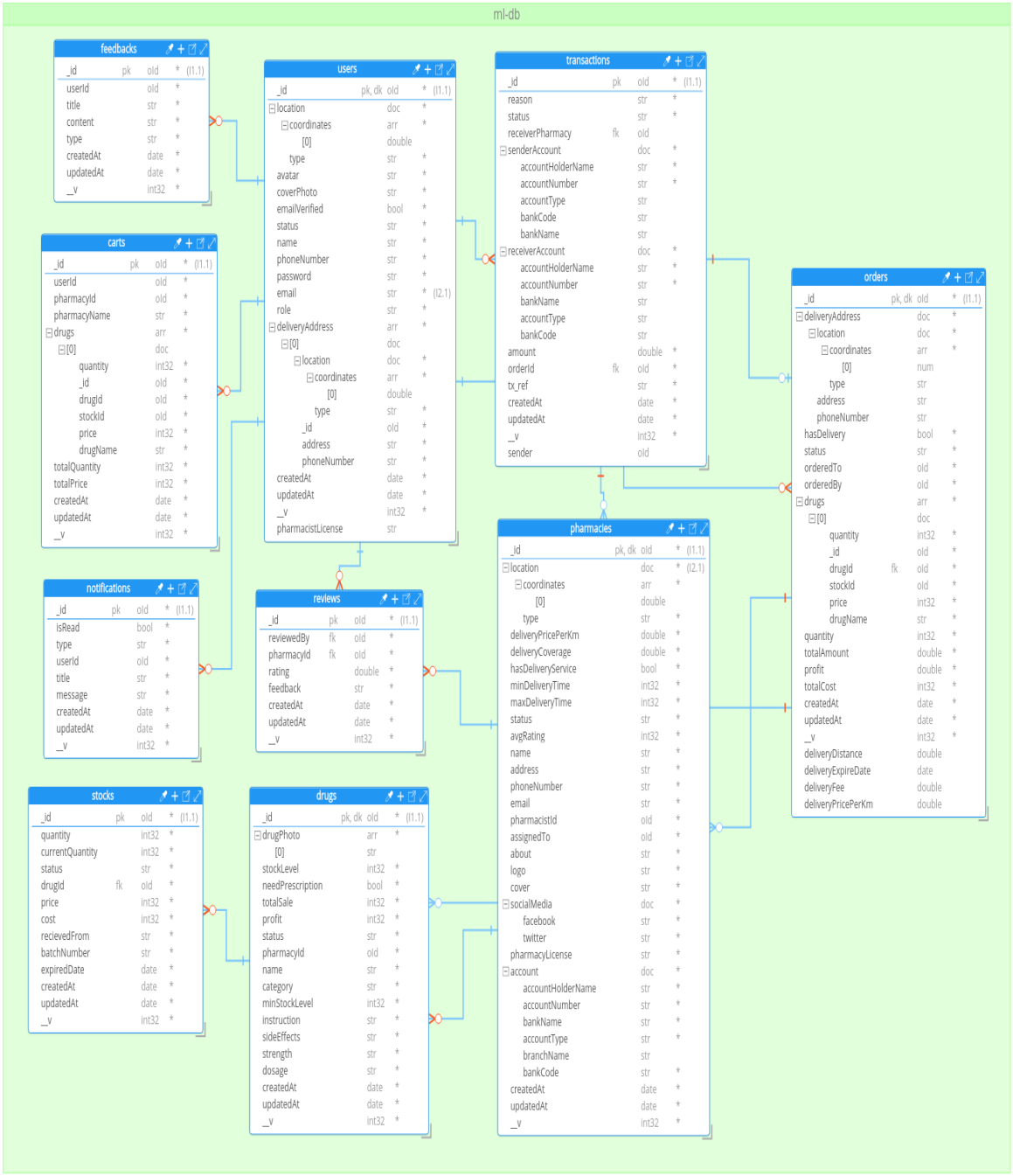


Fig 5.5 persistence data diagram

5.3.6. Access Control and Security

In the Medicine Locator system, diverse user roles are assigned varying levels of access to its features. During the analytical phase, these distinctions are modeled by associating different use cases with specific actors. When crafting the system, access is modeled through the object-oriented framework. This involves determining which items are shared among actors and explicitly specifying how these actors can exercise control over access. Authentication methods are established in alignment with the security requirements of the system, verifying users based on their provided username and password.

Upon successful authentication, the system identifies the user and presents a tailored interface, displaying a window that corresponds to the specific role of that user. A role-based approach is employed to streamline access management, ensuring that each user is granted appropriate permissions based on their designated role within the system.

To articulate and regulate the scope of operations accessible to each actor, an access control table is implemented. This table serves as a simulation of access class control, systematically defining and organizing the permissions associated with each user role. This role-based access model enhances the system's security and efficiency, allowing for a more granular control over user privileges.

Table 5. 1 Access control and security

Role	Resource	Actions	Attributes
Customer	Users	read:own, update:own, create:own	*, !email, !role, !emailVerified, !pharmaciestLicense
	Notifications	read:own, update:own, delete:own	*
	Feedbacks	read:own, create:own, update:own	*
	Drugs	read:any	*, !minStoclLevel, !totalSale, !profit
	Pharmacies	read:any	*
	Transaction	read:own	*
	Reviews	read:any, create:own	*
	Carts	read:own, update:own, create:own, delete:own	*
	Orders	read:own, update:own, create:own	*
	Stocks	read:any	*, !quantity
Pharmaciest	Users	read:own, update:own, create:own	*, !email, !role, !pharmaciestLicense, !emailVerified,

	Notifications	read:own, update:own, delete:own	*	

	Feedbacks	read:own, create:own, update:own	*	
	Transaction	read:own	*	
	Drugs	read:own	*	
	Orders	read:own	*	
	Stocks	read:own	*	
	Pharmacies	read:own, update:own	*	
	Reviews	read:own	*	
Admin	Users	read:own, read:any, update:own, update:any	*, !password, !location, !deliveryAddress, !emailVerified, !address, !email, !role, !pharmaciestLicense	
	Notifications	read:own, update:own, delete:own	*	

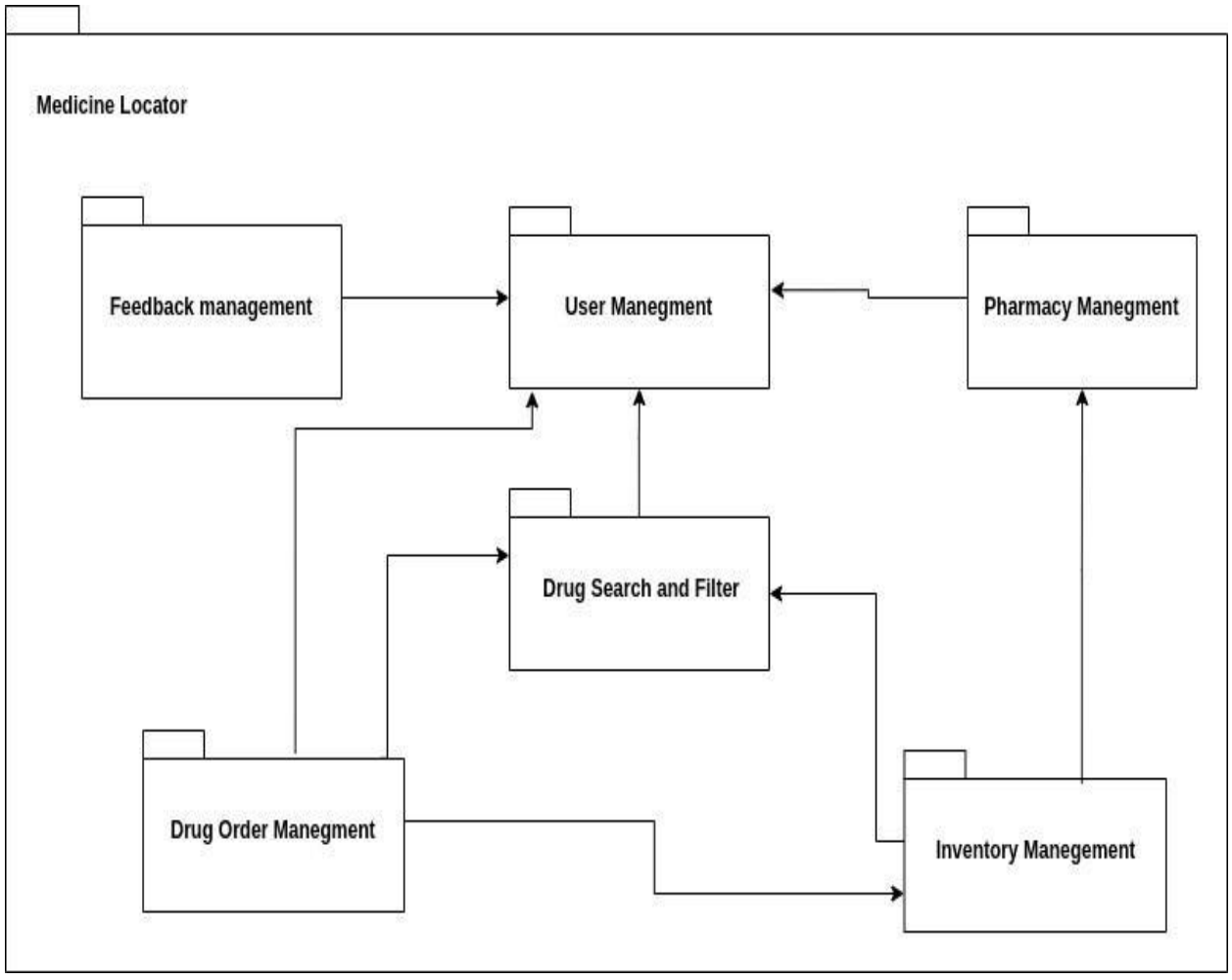
	Feedbacks	read:any, read:own, create:own, update:own, update:any	*
	Pharmacies	read:any, update:any	status
	Transaction	read:any	*, !receiverAccount
	Reviews	read:any	*
SuperAdmin	Users	read:own, read:any, update:own, update:any, create:any, create:own	name, email, password, role, *, !password,!deliveryAddress, !location, !address, !email, !role, !emailVerified, !pharmaciestLicense
	Notifications	read:own, update:own, delete:own	*
	Feedbacks	read:any, read:own, create:own, update:own, update:any	*
	Pharmacies	read:any, update:any	status
	Transaction	read:any	*, !receiverAccount
	Reviews	read:any	*

	RolesPermiss ion s	read:any, update:any, delete:any, create:any	*

5.4. Packages

The package diagram is an abstraction that simplifies the complexity of the Medicine Locator System. It provides a holistic view of the major packages, illustrating their interdependencies and highlighting the broader architecture of the system. Each package encapsulates related elements, fostering a modular and maintainable design.

Fig 5.6 package diagram



5.5 Algorithm Design

Pseudocode for Guest User:

Use Case: Search for Medicines

Algorithm: Search Medicines (medicineName)

Input: medicineName

Output: List of pharmacies with the availability of the medicine

Begin

Enter medicineName in the search bar

Click the search button

If medicines found

Display list of pharmacies with medicine availability

Else

Display message "Medicine not found"

End if

End

Use Case: View Pharmacy Details

Algorithm: View Pharmacy Details (pharmacyID)

Input: pharmacyID

Output: Pharmacy details

Begin

Click on the pharmacy details link

If pharmacyID exists

Display pharmacy details

Else

Display message "Pharmacy not found"

End if

End

Use Case: Register to the System

Algorithm: Register (username, password, fname, lname, email)

Input: username, password, fname, lname, email

Output: Registration success message

Begin

Click on the registration link

Fill the registration form with username, password, fname, lname, email

If registration data is valid and unique

Create a new user account

Display registration success message

Else

Display registration failure message

End if

End

Pseudocode for Customer:

Use Case: Log in or out the System

Algorithm: User Login (username, password)

Input: username, password

Output: Login success or failure

Begin

Enter username and password

Click the login button

If username and password match

Display login success message

Else

Display login failure message

End if

End

Use Case: Rate the Pharmacy

Algorithm: Rate Pharmacy (pharmacyID, rating)

Input: pharmacyID, rating

Output: Rating submitted successfully message

Begin

Click on the "Rate Pharmacy" link

Enter the rating for the pharmacy

If rating is valid

Submit the rating

Display rating submitted successfully message

Else

Display rating submission failure message

End if

End

Use case: Request Refund

Algorithm: Request_refund(customerId, orderId)

Input: customerId, orderId

Output: refund succeed or failed

Begin

If order not delivered on time:

Refund request button is active

Click refund button

The system sends users order and delivery
fee If transaction succeeds: Display
transaction succeed Else:

Display transaction failed
End if
End

Use case: Confirm delivery
Algorithm: Confirm Delivery (orderID)
Input: orderID
Output: Delivery confirmation success or failure

Begin
Click on the "Confirm Delivery" link
If orderID exists and the order is marked as delivered Confirm
the delivery
transfer order and delivery fee to the pharmacy account
Display delivery confirmation success message
Else
Display delivery confirmation failure message
End if
End

Pseudocode for System Administrator:

Use Case: Approve Pharmacy
Algorithm: Approve Pharmacy (pharmacyID)
Input: pharmacyID
Output: Pharmacy approval success or failure

Begin

Click on the "Approve Pharmacy" link

If pharmacyID exists

Approve the pharmacy

Display approval success message

Else

Display approval failure message

End if

End

Use Case: Manage the System

Algorithm: Manage System (actions)

Input: actions

Output: System management success or failure

Begin

Perform system management actions (e.g., update system settings, manage users)

If actions are successful

Display management success message

Else

Display management failure message

End if

End

Pseudocode for Pharmacist:

Use Case: Update Inventory Information

Algorithm: Update Inventory (pharmacyID, medicineID, quantity)

Input: pharmacyID, medicineID, quantity

Output: Inventory update success or failure

Begin
Click on the "Update Inventory" link
If pharmacyID and medicineID exist
Update the inventory with the specified quantity
Display inventory update success message
Else
Display inventory update failure message
End if
End

Use Case: Manage Orders

Algorithm: Manage Orders (orderID, actions)
Input: orderID, actions
Output: Order management success or failure

Begin
Click on the "Manage Orders" link
Perform order management actions (e.g., process order, cancel order)
If actions are successful
Display order management success message
Else
Display order management failure message
End if
End

Use Case: Update Pharmacy Details

Algorithm: Update Pharmacy Details (pharmacyID, details)

Input: pharmacyID, details

Output: Pharmacy details update success or failure

Begin

Click on the "Update Pharmacy Details" button

If pharmacyID exists

Update the pharmacy details with the specified information

If update/addition is successful

Display details update success message

Else

Display details update failure message

End if

End if

End

Use case Reject order

Algorithm: reject order (orderId)

Input:orderId

Output: Rejection success or failure

Begin

Click on the "reject order" button

If orderId exists

Update the status of order to rejected

Refund customer order and delivery fee

If transaction is successful

Display order rejection success message

Else

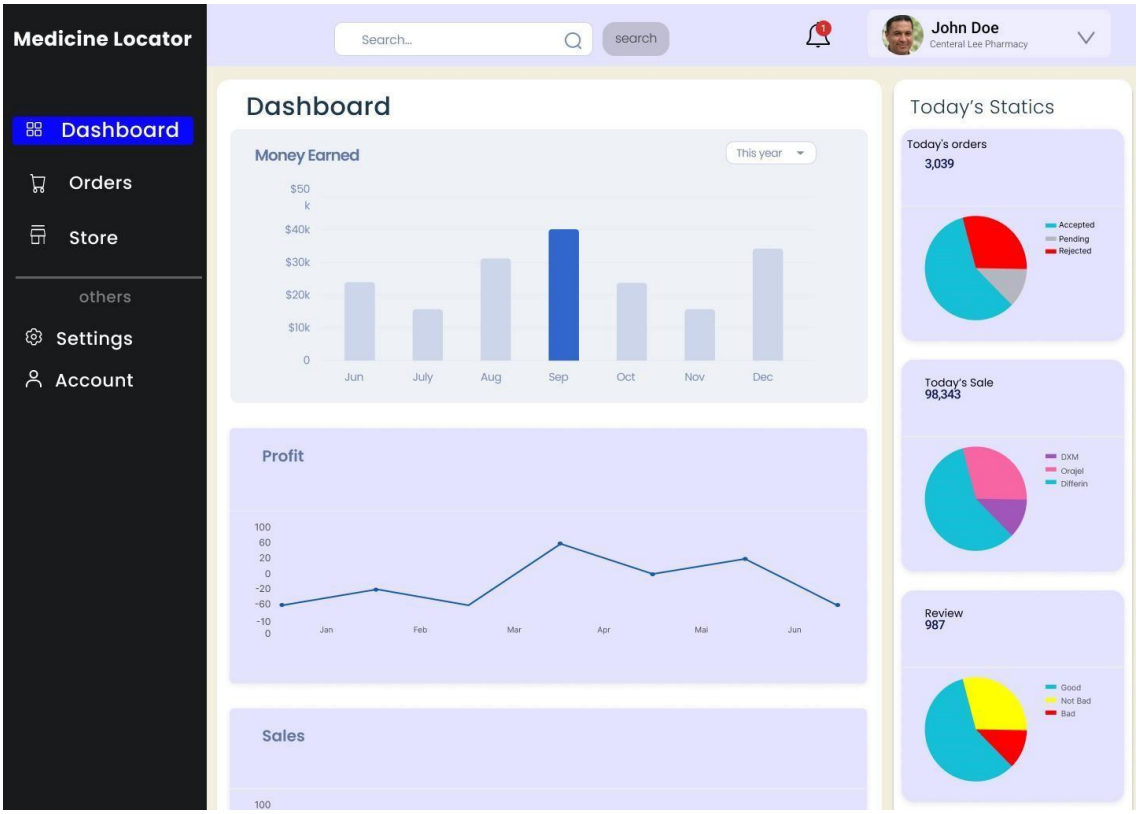
Display order failure message

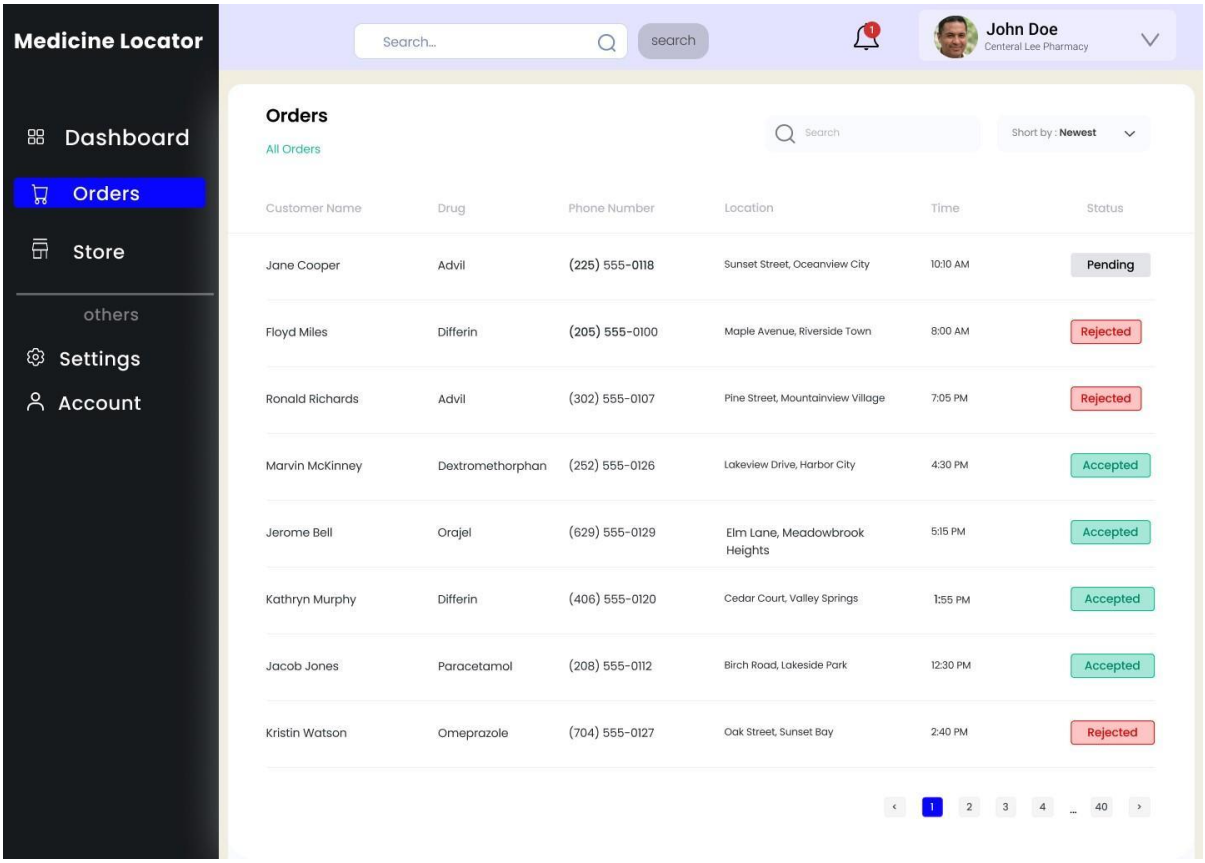
End if

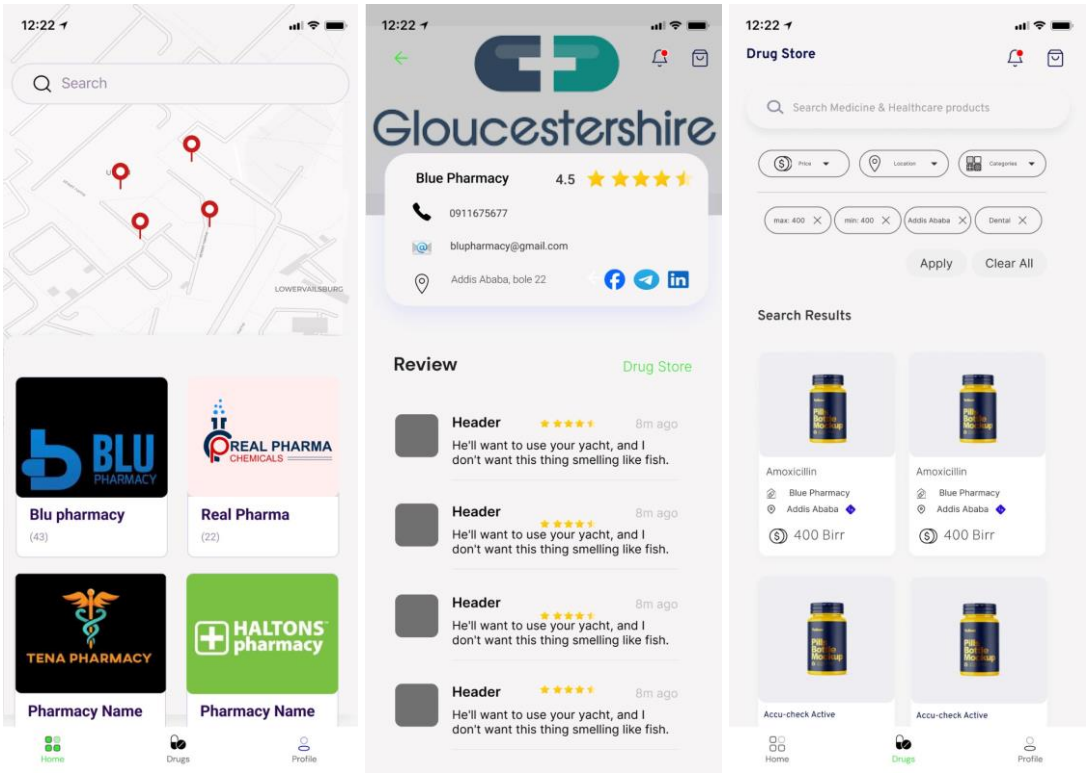
End if

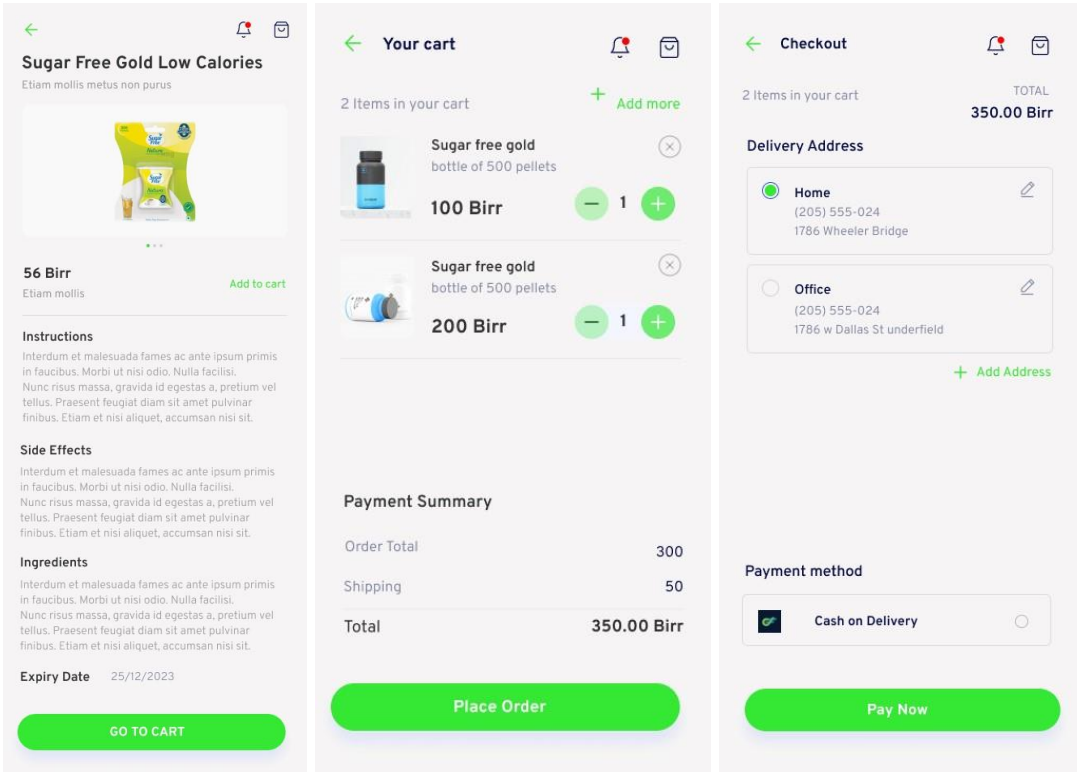
End

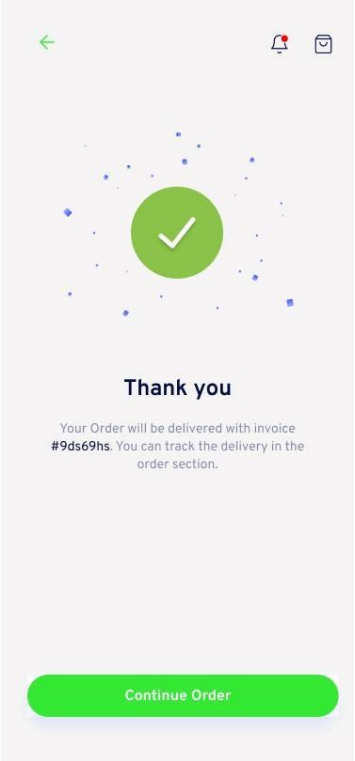
5.6. User Interface Design




Medicine Locator
Search...
search
John Doe
Centeral Lee Pharmacy
Dashboard
Orders
Store
others
Settings
Account
Orders
All Orders
Search
Short by : Newest
Customer Name
Drug
Phone Number
Location
Time
Status
Jane Cooper
Advil
(225) 555-0118
Sunset Street, Oceanview City
10:10 AM
Pending
Floyd Miles
Differin
(205) 555-0100
Maple Avenue, Riverside Town
8:00 AM
Rejected
Ronald Richards
Advil
(302) 555-0107
Pine Street, Mountainview Village
7:05 PM
Rejected
Marvin McKinney
Dextromethorphan
(252) 555-0126
Lakeview Drive, Harbor City
4:30 PM
Accepted
Jerome Bell
Orajel
(629) 555-0129
Elm Lane, Meadowbrook Heights
5:15 PM
Accepted
Kathryn Murphy
Differin
(406) 555-0120
Cedar Court, Valley Springs
1:55 PM
Accepted
Jacob Jones
Paracetamol
(208) 555-0112
Birch Road, Lakeside Park
12:30 PM
Accepted
Kristin Watson
Omeprazole
(704) 555-0127
Oak Street, Sunset Bay
2:40 PM
Rejected
1
2
3
4
40


12:22
Search
Blu pharmacy
(43)
Real Pharma
(22)
Pharmacy Name
Pharmacy Name
Home
Drugs
Profile
Gloucestershire
Blue Pharmacy
4.5
0911675677
blupharmacy@gmail.com
Addis Ababa, bole 22
Review
Drug Store
Header
8m ago
He'll want to use your yacht, and I don't want this thing smelling like fish.
Drug Store
Search Medicine & Healthcare products
Apply
Clear All
Search Results
Amoxicillin
Blue Pharmacy
Addis Ababa
400 Birr
Accu-check Active


Sugar Free Gold Low Calories
Etiam mollis metus non purus
56 Birr
Etiam mollis
Add to cart
Instructions
Side Effects
Ingredients
Expiry Date 25/12/2023
GO TO CART
Your cart
2 Items in your cart
Add more
Sugar free gold
bottle of 500 pellets
100 Birr
200 Birr
Payment Summary
Order Total 300
Shipping 50
Total 350.00 Birr
Place Order
Checkout
2 Items in your cart
TOTAL
350.00 Birr
Delivery Address
Home
(205) 555-024
1786 Wheeler Bridge
Office
(205) 555-024
1786 w Dallas St underfield
Add Address
Payment method
Cash on Delivery
Pay Now


Thank you
Your Order will be delivered with invoice #9ds69hs. You can track the delivery in the order section.
Continue Order

CHAPTER SIX

IMPLEMENTATION AND TESTING

6.1 INTRODUCTION

In this pivotal phase, we breathe life into the blueprint of our design. The essence lies in translating the meticulous physical design specification into a robust, efficient codebase. Our mission: to forge a tangible Minimum Viable Product (MVP), ready to navigate the real-world terrain. Here, we meticulously craft each line of code, laying the groundwork for seamless unit and integration testing. It's the crucible where ideas crystallize into action, where theory meets reality, and where innovation takes its first steps forward.

6.2 IMPLEMENTATION OF THE DATABASE

In our database implementation, we leveraged the MongoDB document-based database system. This strategic decision was informed by its exceptional compatibility with React and Node.js, essential components of the acclaimed MERN stack—the forefront of contemporary web development. MongoDB distinguishes itself with its innate flexibility in schema definition, a departure from the rigid structures of SQL-based databases. This flexibility not only streamlines development but also facilitates seamless adaptation to evolving project needs. Moreover, MongoDB's horizontal scalability offers a distinct advantage over traditional SQL databases, ensuring optimal performance as our application grows in complexity and scale.

- All of mongdb collection and their relationship

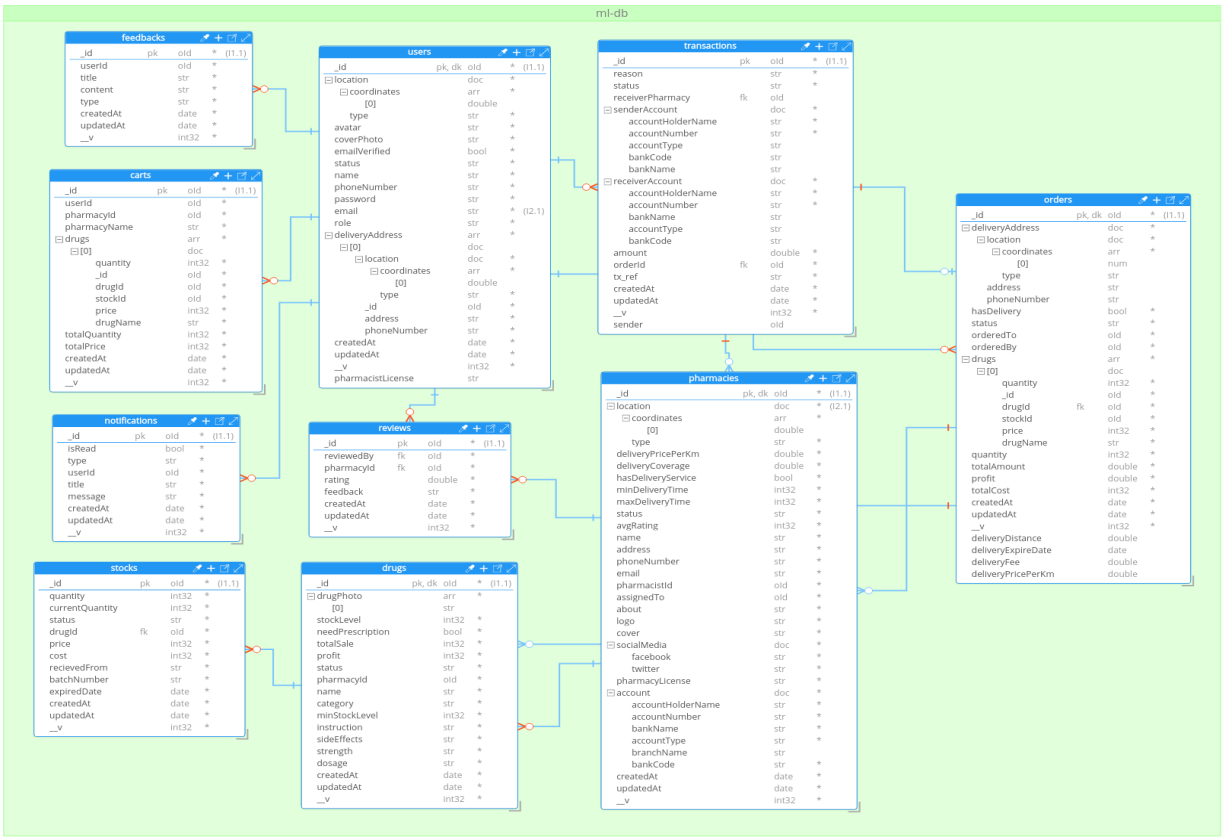


Fig 6.1 implementation of the database

- In MongoDB, the _id field serves as the default index for each document. Additionally, we have implemented an index on the "pharmacies" collection, particularly on the "location" field. These indices are instrumental in facilitating efficient location-based queries, enabling seamless calculation of distances relative to a user's location. Leveraging MongoDB's aggregation framework, specifically the $geoNear pipeline stage, this index empowers us to swiftly locate nearby pharmacies and perform geospatial operations with precision and speed.

- We have integrated MongoDB Atlas scheduled triggers to execute specific tasks at predefined intervals. These scheduled triggers operate at designated times, ensuring timely execution of essential functions. Below is a list of some scheduled trigger Implementation:

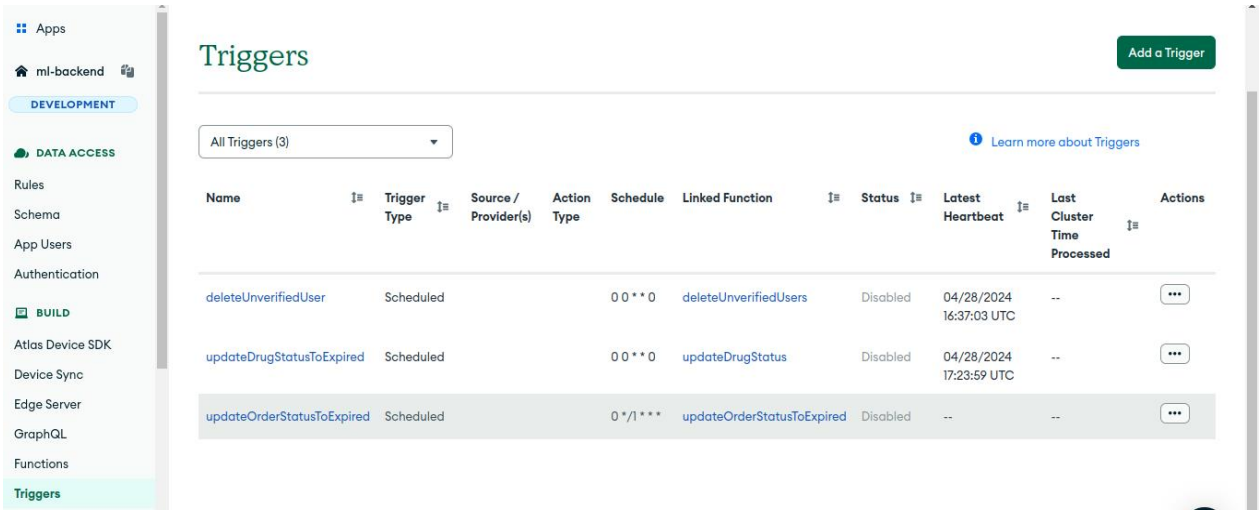


Fig 6.2 database scheduled trigger Implementation

- MongoDB Atlas offers default backup and security configurations, which are integral components of its service. These features are meticulously designed to provide users with peace of mind regarding data protection and system integrity. This inherent advantage eliminates the need for extensive manual configuration, freeing us to focus on other critical aspects of projects.

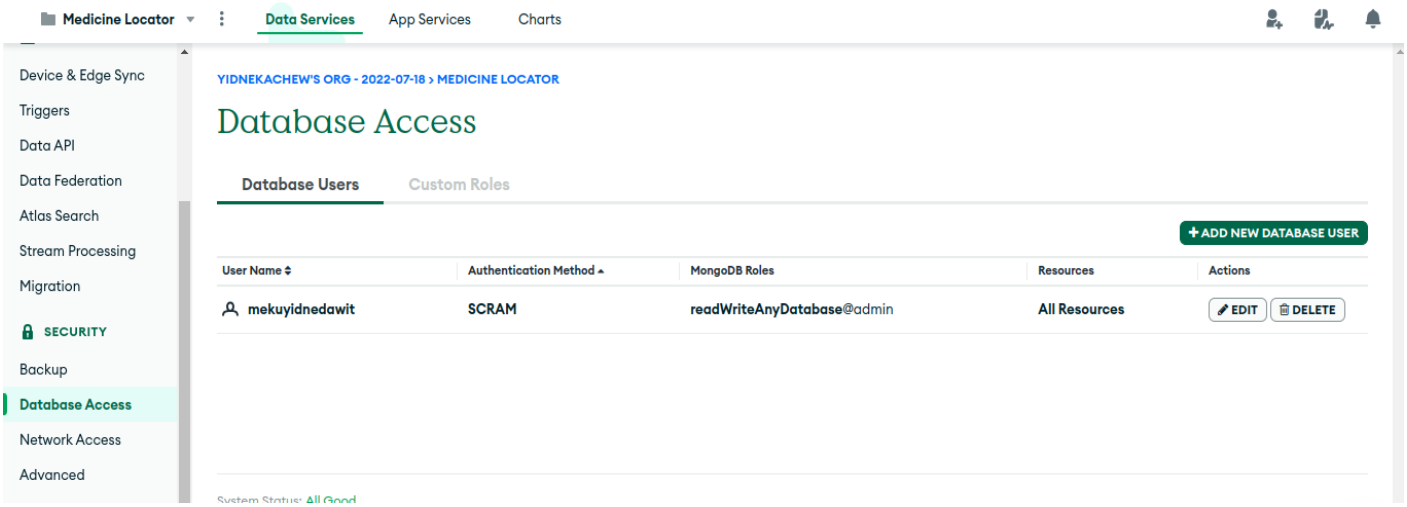


Fig 6.3 database security configurations

6.4 IMPLEMENTATION OF THE CLASS DIAGRAM

Our backend follows the MVC (Model-View-Controller) architecture, employing Mongoose models to structure our database schema and facilitate interactions with MongoDB. Using controller functions in conjunction with Express.js, we manage API endpoint requests, executing application logic and handling CRUD operations. Each class defined in our class diagram has its corresponding model folder, comprising schema.js, method.js, statics.js, and index.js files. These files collectively encapsulate the class functionalities, ensuring modularity and ease of maintenance. Below is the folder structure housing all our models.

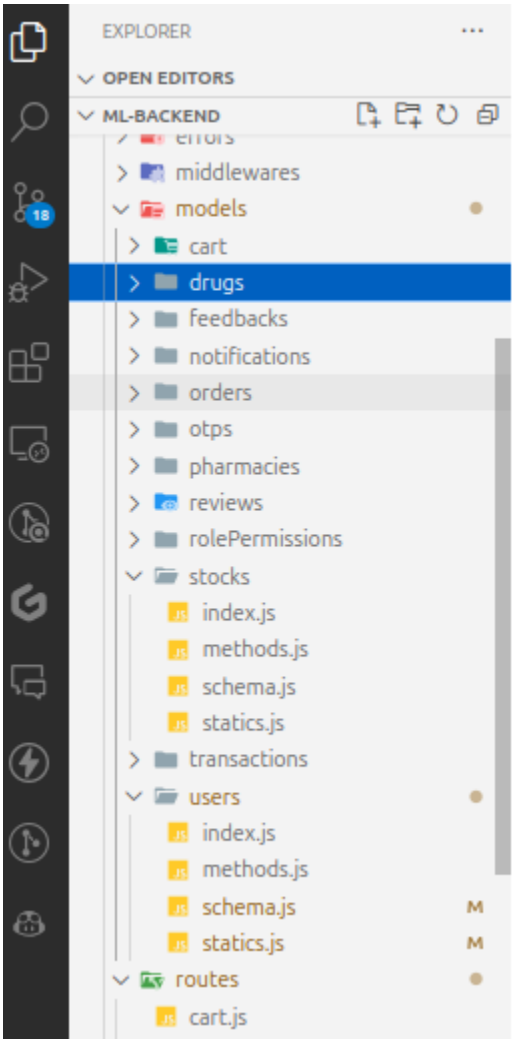


Fig 6.4 implementation of class diagram

- here is an example user model schema

```
You, 4 hours ago | 1 author (You)
import mongoose from 'mongoose';

const userSchema = new mongoose.Schema(
  {
    name: { type: String, required: true },
    phoneNumber: { type: String },
    password: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    address: {
      type: String,
    },
    emailVerified: { type: Boolean, default: false, required: true },
    role: {
      type: String,
      enum: ['admin', 'pharmacist', 'customer', 'superAdmin'],
      required: true,
    },
    status: {
      type: String,
      enum: ['active', 'inactive', 'deactivated'],
      default: 'active',
    },      You, 4 hours ago • Uncommitted changes
    avatar: {
      type: String,
      default: 'https://fakeimg.pl/150x150/bdbdbd/ffffff?text=avatar&font=noto',
    },
    coverPhoto: {
      type: String,
      default:
        'https://fakeimg.pl/400x200/bdbdbd/ffffff?text=Cover+Photo&font=noto',
    },
    location: {
```

Fig 6.5 user model schema

- here an example user controller

```
You, 5 days ago | 1 author (You)
/* eslint-disable import/no-extraneous-dependencies */      You, 5 months ago • [feat]: filterPharmacy, pharmacy detail and fil…
import httpStatus from 'http-status';
import User from '../models/users';
import APIError from '../errors/APIError';
import Otp from '../models/otps';
import { generateOtp } from '../utils';

export const sendOTP = async (req, res, next) => {
  const { email, type } = req.body;
  const otp = generateOtp(6);
  const otpPayload = { otp, email, type };
  try {
    const existingUser = await User.findOne({ email });

    if (!existingUser) {
      throw new APIError('email not found', httpStatus.NOT_FOUND, true);
    }
    const message = await Otp.createOtp(otpPayload);
    res.status(httpStatus.OK).json(message);
  } catch (local var) error: any
    next(error);
  }
};

export const verifyOTP = async (req, res, next) => {
  const { email, code } = req.body;
  try {
    const data = {
      email,
      otp: code,
    };
    const message = await Otp.verifyOtp(data);
```

Fig 6.6 user controller

6.5 CONFIGURATION OF THE APPLICATION SERVER

Our backend application server, constructed with the Node.js Express framework, boasts remarkable configurational versatility. It effortlessly adapts to commence operation on any designated IP address and port number, thus accommodating a wide array of deployment scenarios and networking prerequisites. In accordance with our system architecture, which adheres to the three-tier model, the frontend, backend, and MongoDB components each reside on their respective servers, ensuring a meticulously organized infrastructure.

Here is the script responsible for spinning up our Node.js Express server and connect it with mongodb:

```
...
/* eslint-disable import/first */
import path from 'path';
// Initiate app root
global.appRoot = path.resolve(path.resolve());
import passport from 'passport';
import cors from 'cors';
import * as environments from './config/environments';
import connectToDb from './config/mongoose';
import app from './config/express';
import passportInit from './config/passport';

app.use(cors());
passportInit(passport);

const start = async () => {
  if (!module.parent) {
    await connectToDb();

    app.listen(environments.port, () => {
      // eslint-disable-next-line no-console
      console.log(
        `[${environments.nodeEnv}] Server running on localhost:${environments.port}`
      );
    });
  }
};
start();

export default app;
```

Fig 6.7 configuration of the Application server

6.6 CONFIGURATION OF APPLICATION SECURITY

- In ensuring robust application security, we've implemented two crucial components: bcrypt and JSON Web Tokens (JWT).

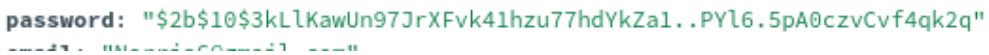
- Firstly, bcrypt, a widely acclaimed npm package, is employed for encryption and decryption of sensitive user information, such as passwords. It employs a one-way hashing algorithm, making it exceptionally secure for safeguarding user credentials. When a user registers or updates their password, bcrypt securely hashes it before storing it in the database. During login attempts, bcrypt compares the hashed password with the stored hash, verifying the user's identity without ever exposing the actual password.

- Here's an example line of code demonstrating how we hash user passwords during signup using bcrypt:

```
const hashedpassword = await bcrypt.hash(password, 10);
```

This line of code is taking a plain text password, hashing it 10 times using bcrypt, and storing the resulting hash in the hashedpassword variable. This hashed password can then be stored into the database, which is much safer than storing the plain text password.

here encrypted user password saved in the database



- Here's an example line of code demonstrating how we use bcrypt compare current user password and the hash password in the database during user login:

```
const passwordMatch = await bcrypt.compare(password, user.password);
```

This line of code is comparing a plain text password with a hashed password and storing the result of the comparison boolean in the passwordMatch variable.

- Secondly, we utilize JSON Web Tokens (JWT) for user session management. Upon successful authentication, the server generates a JWT containing essential user information, such as their role and user ID. This token is digitally signed using a secret key to ensure its authenticity and integrity. The token also includes an expiration date, providing an additional layer of security.

- Here is an example script that generates an access token and refresh token if the passwordMatch variable is true.

```
const passwordMatch = await bcrypt.compare(password, user.password);
if (passwordMatch) {
  const token = generateJwtAccessToken(user._id);
  const newRefreshToken = generateJwtRefreshToken(user._id);
  const cleanUser = user.clean();
  cleanUser.accessToken = token;      You, 3 months ago • Refactor log
  cleanUser.refreshToken = newRefreshToken;
  return cleanUser;
```

- Throughout the user's session, the JWT is included in the HTTP header of subsequent requests and we use another npm package which is called passport.js which is a middleware that intercept the jwt token from the request header and verify the validity of the jwt token.

Here is an example code:

```
3   import httpStatus from 'http-status';
4   import passport from 'passport';
5   import ac from '../config/accesscontrol';
6   import APIError from '../errors/APIError';
7
8   const authenticateJwt = (req, res, next) => {
9     if (req.headers && !req.headers.authorization) {
10      const missingTokenError = new APIError(
11        'Token not found! Provide credential',
12        httpStatus.BAD_REQUEST
13      );
14      return next(missingTokenError);
15    }
16
17    return passport.authenticate(
18      'jwt',
19      { session: false },
20      (error, user, message) => {
21        if (error || !user) {
22          throw new APIError(message, httpStatus.UNAUTHORIZED, true);
23        }
24
25        req.user = user.clean();
26        return next();      You, 5 months ago • model schema setup
27      }
28    )(req, res, next);
29  };
```

Fig 6.8 JWT authentication middleware

and we used this middleware in all API endpoint that require authentication for example:

```
router.put('/:orderId/refund', import authenticateJwt
router.put('/:orderId/reject', authenticateJwt, rejectOrderController);
router.put('/:orderId/accept', authenticateJwt, acceptOrderController);
router.put('/:orderId/extend', authenticateJwt, extendOrderController);

export default router;
```

- We've also implemented backend validation using the Express Validator library. This ensures thorough validation of all data originating from the frontend request object, including req.body, req.params, and req.query and used as a middleware.
- For example:

```
export const signUpUserValidator = () => [
  body('name')
    .isString()
    .isLength({ min: 5, max: 40 })
    .withMessage('name is required and must be at least 5 characters'),
  body('phoneNumber')
    .optional()
    .isString()
    .custom((phone) => {
      const regex = /((^(2517|2519|07|09)\d{3})-?\d{5})/;
      return regex.test(phone);
    })
    .isLength({ min: 10, max: 12 })
    .withMessage(
      'Phone number is required and must be valid. valid codes are 2517/07 or 2519/09'
    ),
  body('password')
    .isString()
    .isLength({ min: 4, max: 60 })
    .withMessage(
      'Password should be at least 4 cahracters and not greater than 60'
    ),
  body('role')
    .isString()
    .custom((val) =>
      ['admin', 'pharmacist', 'customer', 'superAdmin'].includes(
        val.toLowerCase()
      )
    )
    .withMessage('A valid role is required'),
  body('email').isEmail().withMessage('Email is required'),
];
```

Fig 6.8 implementation backend validation

here how we used this validation middleware:

Fig 6.8 implementation of validation middleware

6.8 TESTING

During development, we utilize the Thunder Client manual API endpoint testing extension for Visual Studio Code. This extension streamlines the testing process during development, allowing us to quickly send requests to backend endpoints and inspect responses directly within the code editor.

For example:

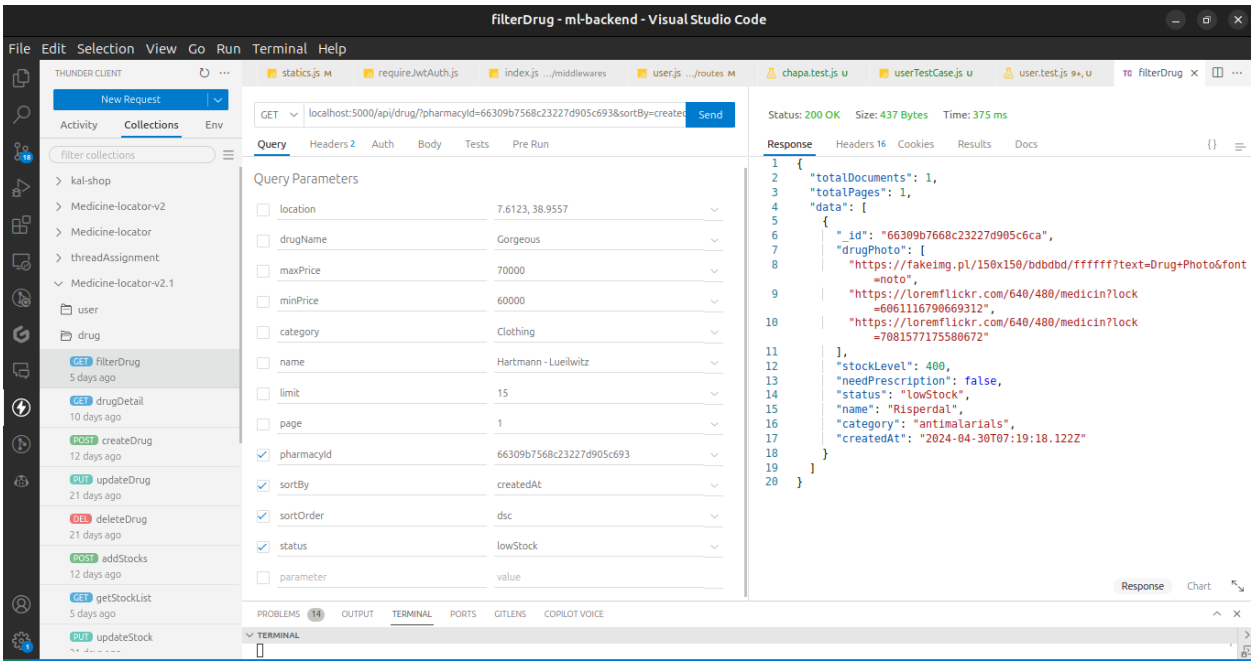


Fig 6.8 Thunder Client manual API endpoint testing

After development we have meticulously set up automated integration and unit testing for our backend application using the Mocha and Chai testing frameworks, along with the Chai-HTTP plugin. These tools allow us to thoroughly assess the functionality of our backend components, ensuring robustness and reliability.

Mocha is a flexible JavaScript test framework that runs on both Node.js and in the browser. It provides a simple, yet powerful, syntax for organizing and executing tests, allowing developers to easily define test suites and individual test cases.

Chai is an assertion library that works seamlessly with Mocha to enhance test readability and maintainability. It offers a variety of assertion styles, including BDD (Behavior-Driven Development) and TDD (Test-Driven Development), enabling developers to express test expectations in a clear and concise manner.

Chai-HTTP is an extension of Chai that provides a set of utilities for testing HTTP endpoints. It simplifies the process of sending HTTP requests and asserting responses within test cases, making it ideal for integration testing of RESTful APIs.

here is an example integration testing test case:

__test__ > routes > user > userTestCase.js > invalidEmail

```
1   const validNewUser = {
2     name: 'John Doe',
3     email: 'john.doe@example.com',
4     password: 'Password@123',
5     phoneNumber: '251912345678',
6     role: 'customer',
7   };
8
9   const validNewPharmacist = {
10    name: 'Alice Smith',
11    email: 'alice.smith@example.com',
12    password: 'Pass1234',
13  };
14
15  const validNewAdmin = {
16    name: 'Admin User',
17    email: 'admin@gmail.com',
18    role: 'admin',
19  };
20
21  const missingName = {
22    email: 'john.doe@example.com',
23    password: 'Password@123',
24  };
25
26  const missingEmail = {
27    name: 'John Doe',
28    password: 'Password@123',
29  };
30
```

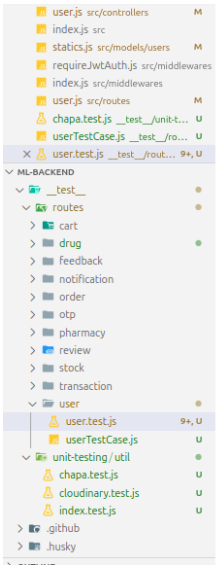
Fig 6.8 integration testing test cases

here is an example integration testing for user endpoints:

```
8 > import {…
35  } from './userTestCase';
36
37  chai.use(require('chai-http'));
38
39  const { request, expect } = chai;
40
41  describe('User Routes Test', () => {
42    before((done) => {
43      connectToDb().then(() => {
44        done();
45      });
46    });
47
48    describe('# [POST] /api/user/signUp', () => {
49      it('Should create a new user with role customer send otp to users email', async () => {
50        const response = await request(app)
51          .post('/api/user/signUp')
52          .send(validNewUser);
53
54        const { success } = response.body;
55
56        expect(response).to.have.status(httpStatus.OK);
57        expect(response.body).to.be.an('object');
58        expect(success).to.equal(true);
59      });
60
61      it('Should return a status of 409 [Conflict] when an existing E-mail is used to sign up', async () => {
62        const response = await request(app)
63          .post('/api/user/signUp')
64          .send(duplicateEmail);
65        expect(response).to.have.status(httpStatus.CONFLICT);
66      });
67
```

Fig 6.8 integration testing for user endpoints

here is an example test result when we run yarn test command in terminal to run all test case is our __test__ folder:




```
POST /api/user/signUp 409 93.518 ms - 35
      ✓ Should return a status of 409 [Conflict] when an existing E-mail is used to sign up (99ms)
POST /api/user/signUp 400 2.136 ms - 230
      ✓ Should return a status of 400 [BAD REQUEST] when email type is invalid
POST /api/user/signUp 400 4.771 ms - 288
      ✓ Should return a status of 400 [BAD REQUEST] when name is not provided
POST /api/user/signUp 400 3.043 ms - 310
      ✓ Should return a status of 400 [BAD REQUEST] when password is not provided
POST /api/user/signUp 400 3.309 ms - 310
      ✓ Should return a status of 400 [BAD REQUEST] when password is length is too short
    # [POST] /api/user/login
POST /api/user/login 200 94.471 ms - 835
      ✓ Should login user (100ms)
POST /api/user/login 401 91.434 ms - 46
      ✓ Should return a status of 401 [UNAUTHORIZED] when login credentials are invalid (95ms)
    # [POST] /api/user/refresh-token
POST /api/user/refresh-token 401 4.749 ms - 53
      ✓ Should return a status of 401 [UNAUTHORIZED] when refresh token is invalid
    # [POST] /api/user/reset-password
POST /api/user/reset-password 401 92.799 ms - 39
      ✓ should return 401 [UNAUTHORIZED] when there is not valid otp (100ms)
POST /api/user/reset-password 400 1.478 ms - 141
      ✓ Should return a status of 400 [BAD REQUEST] when password is invalid
    # [POST] /api/user/pharmacist
POST /api/user/pharmacist 400 3.805 ms - 43
      ✓ Should return a status of 400 [BAD REQUEST] when file is not uploaded
    # [POST] /api/user/admin
POST /api/user/admin 400 2.820 ms - 131
      ✓ Should return a status of 400 [BAD REQUEST] when role is invalid
    # [GET] /api/user
GET /api/user 200 15.222 ms - 175
      ✓ Should get all users


  14 passing (820ms)

Done in 10.67s.
➜  ml-backend git:(staging) ✗
```

Fig 6.8 example test result

here is chapa payment method functionality testCase:

__test__ > unit-testing > util > chapatestCase.js > invalidAmount

```
/* eslint-disable camelcase */

import { v4 as uuidv4 } from 'uuid';

const validPaymentDetails = {
  amount: '1000', // Example amount in cents
  first_name: 'John',
  last_name: 'Doe',
  phone_number: '0900123456',
  currency: 'ETB',
  tx_ref: uuidv4(),
};

const missingReqTx_Ref = {
  ...validPaymentDetails,
  tx_ref: '',
};

const invalidCurrency = {
  ...validPaymentDetails,
  currency: 'AAA', // Invalid currency
};

const invalidAmount = {
  ...validPaymentDetails,
  amount: -100, // Negative amount
};

const validTransactionRef = validPaymentDetails.tx_ref;

const invalidTransactionRef = 'TX987654321';

const nonExistentTransactionRef = 'TX000000000';

```

Here's an example of unit testing for the Chapa payment initialization function:

```
const { expect } = chai;

describe('Chapa payment method functionality Test', () => {
  describe('Test initializePayment function', () => {
    it('Should return a string url that can display chapa payment', async () => {
      const response = await initializePayment(validPaymentDetails);
      expect(response.data.checkout_url).to.be.a('string');
      expect(response.status).to.equal('success');
    });

    it('Should throw an error if amount is invalid', async () => {
      try {
        await initializePayment(invalidAmount);
      } catch (error) {
        expect(error).to.equal(
          'Error: AxiosError: Request failed with status code 400'
        );
      }
    });
  });
});
```

While we have not conducted exhaustive testing of the backend application due to time constraints, we assure that all testing tools and configurations are properly set up. These tools will serve as valuable resources for the maintenance team, enabling them to thoroughly assess the backend's functionality and guarantee its stability and performance in the future.

CHAPTER SEVEN 7

CONCLUSION AND RECOMMENDATION

7.3. CONCLUSION

This project, the Medicine Locator System, aimed to address the challenges of medicine availability and accessibility in Ethiopia. The system was designed to provide users with a convenient platform to search for and locate medicines within their vicinity, ultimately improving healthcare access. Through meticulous planning, analysis, and development, the project successfully achieved its objectives. The system's user-friendly interface, comprehensive database, and real-time updates ensure that users can efficiently find the medicines they need. Moreover, the system's potential impact on the Ethiopian healthcare landscape is significant, as it bridges the gap between patients and pharmacies, leading to improved medication adherence and overall health outcomes.

7.4. RECOMMENDATION

After thorough evaluation and consideration of the project's scope, user needs, and critical success factors, we believe that the Medicine Locator System is an optimal choice to fulfill our objectives.

As noted, our project's scope is presently limited to a defined set of features, with the potential for future expansions to enhance system functionality. Therefore, it is imperative to select a software solution noted that offers flexibility for scalability and integration with existing healthcare systems. Medicine Locator System demonstrates versatility in accommodating such future developments, ensuring adaptability and longevity for our project.

Furthermore, we acknowledge that the success of our project hinges on several key factors, including user adoption, seamless integration with healthcare infrastructure, and the availability of requisite resources. Medicine Locator System excels in facilitating user engagement through its intuitive interface and comprehensive support services, thereby promoting widespread adoption within our target user base.

Regarding the speed and availability of drug delivery, we recognize the pivotal role of pharmacies in this process. While our system does not directly handle drug delivery, we encourage that pharmacies utilize a Point of Sale (POS) System for real-time registration of sold drugs. This integration ensures efficient tracking of inventory and expedites the delivery process, thereby enhancing overall customer satisfaction.

To optimize the user experience, we propose implementing features within the Medicine Locator System that enable users to verify medicine availability and pharmacy hours, facilitating informed decision-making. Additionally, users should be encouraged to provide feedback, enabling iterative improvements to the system and ensuring alignment with evolving user preferences and requirements.

8. REFERENCES

[1] I. Sommerville, Software Engineering, 10th ed. Boston, MA: Pearson, 2020.

[2] Ethiopian Food and Drug Administration, "Guideline for Medicinal Product Information. "[Online]. Available:http://www.efda.gov.et/publication/guideline-for-medicinal-product-information/

[3] Lucidchart Blog. (n.d.). [Online]. Available: https://www.lucidchart.com/blog.

9. APPENDICES

Appendix I: Interview and Questionnaires

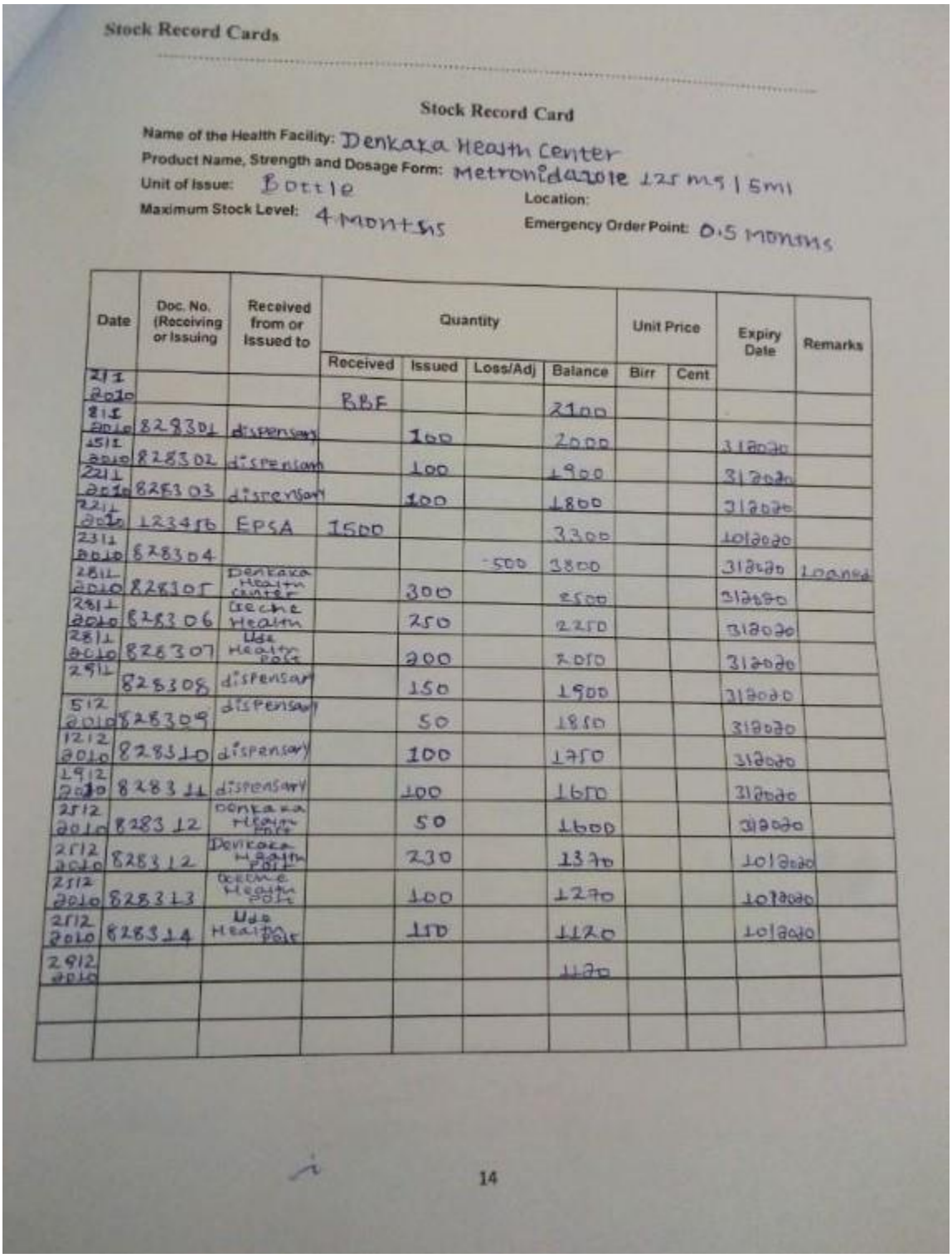
The questionnaire aims to gather information about your current workflow, challenges, and requirements related to medication management, inventory tracking, customer interactions, and regulatory compliance in pharmacies.

- Can you describe your current workflow for managing medication inventory and fulfilling prescriptions?
- What are the main challenges or pain points you encounter in your current system?
- How do you currently track medication availability and inventory levels?

- What specific features would you like the system to have for tracking medication inventory?
- How do you categorize and organize medications in your pharmacy?
- Are there any specific requirements for managing prescription medications versus over-the-counter medications?
- What information do customers typically inquire about when looking for a specific medication?
- How do you currently assist customers in locating medications within your pharmacy?
- Are there any specific features or capabilities you envision for a medication locator system?
- How do you currently handle customer inquiries about medication availability and pricing?
- Do you offer any services for refilling prescriptions or providing medication reminders to customers?
- Are there any additional features you would like to provide to enhance the customer experience?
- Are there any existing systems or software solutions that the new system needs to integrate with (e.g., electronic health records, billing systems)?
- What are your preferences regarding technology platforms and compatibility requirements for the new system?
- What security measures are important to you for protecting patient information and ensuring compliance with healthcare regulations?
- Are there any specific regulatory requirements or standards that the system needs to adhere to?
- What are the specific licensing requirements and regulations for opening and operating a pharmacy in your jurisdiction?
- Are there any specific regulatory bodies or agencies that oversee pharmacy licensing and compliance?
- What steps are involved in obtaining the necessary licenses and permits to open a pharmacy, and what is the timeline for each step?

- What documentation or paperwork is required to apply for pharmacy licensing and permits?
- Are there any specific forms, applications, or supporting documents that need to be submitted as part of the licensing process?

Appendix II: Existing System Forms and Reports



Stock Record Cards

Stock Record Card

Name of the Health Facility: Denkara Health Center
Product Name, Strength and Dosage Form: Metronidazole 125 mg | 5ml
Unit of Issue: Bottle
Location:
Maximum Stock Level: 4 Months
Emergency Order Point: 0.5 Months

Date	Doc. No. (Receiving or Issuing	Received from or Issued to	Quantity				Unit Price		Expiry Date	Remarks
			Received	Issued	Loss/Adj	Balance	Birr	Cent		
2/1 2010			BBF			2100				
8/1 2010	829301	dispensary		100		2000			31/2030	
15/1 2010	828302	dispensary		100		1900			31/2030	
22/1 2010	828303	dispensary		100		1800			31/2030	
22/1 2010	123456	EPSA	1500			3300			10/2030	
23/1 2010	828304				500	3800			31/2030	Loaned
28/1 2010	828305	Denkara Health center		300		3500			31/2030	
29/1 2010	828306	Creche Health		250		2250			31/2030	
28/1 2010	828307	Ude Health Post		200		2050			31/2030	
29/1	828308	dispensary		150		1900			31/2030	
5/2 2010	828309	dispensary		50		1850			31/2030	
12/2 2010	828310	dispensary		100		1750			31/2030	
19/2 2010	828311	dispensary		100		1650			31/2030	
25/2 2010	828312	Denkara Health Post		50		1600			31/2030	
25/2 2010	828312	Denkara Health Post		230		1370			10/2030	
25/2 2010	828313	Creche Health Post		100		1270			10/2030	
25/2 2010	828314	Ude Health Post		150		1120			10/2030	
29/2 2010						1120				

14