



Nada Hamdy AbdAllah

Noha Mohamed Hafez

Rehab Mahmoud Mohamed

Rehab Mahmoud Hassan

Nader Erian Noshy

Amir Girgis Sedkey

Ahmed Elhosienny Mohamed Kamel

Supervised by: Dr.Esraa Elhariri
Dr.Asmaa Hashem

June 25, 2024

Acknowledgement

This project could not be done without God's permission, and no success could be gained without his mercy. We would like to express my sincere gratitude to our supervisors Dr. Esraa M. Elhariri & Dr. Asmaa Hashem for providing their invaluable guidance, comments and suggestions throughout the course of the project, never accepting less than our best efforts. Also we want to thank Eng. Nessma Mohamed, Eng. Fekry Muharram and Eng. Mahmoud Mostafa Sakr for their support and great effort in embedded system part. We thank them all.

Abstract

In recent decades, biomedical and smart healthcare research has experienced significant growth, generating vast amounts of data that exceed human analytical capacity. The complexity of managing this data necessitates automation to effectively extract, represent, and interpret information.

Nabd is an intelligent application designed to address the challenges faced by patients, particularly the elderly and those with strict medication regimens, in managing their medications.

The primary objectives of **Nabd** include facilitating medication management by storing and tracking patients' medications, sending timely reminders, enabling patient supervision, and recommending alternative medications to avoid drug interactions.

The methodology employs deep learning algorithms and leverages two comprehensive datasets to identify potential interactions between drugs and suggest safer alternatives, ensuring patient safety and adherence to prescribed treatments. The system's capability to automate and streamline medication schedules and recommend alternative medications provides a holistic approach to patient care, reducing the burden on both patients and caregivers. **Nabd** demonstrates the potential of advanced technologies in enhancing healthcare outcomes by offering a reliable, user-friendly solution for medication management and patient supervision.

Contents

Acknowledgement 2

Abstract 3

1 Introduction..... 10

1.1 Overview 10

1.2 Problem definition 10

1.3 Project Motivation 11

1.4 Project Objectives 11

1.5 Organization 11

2 Project Planning 12

2.1 Introduction..... 12

2.2 Project Plan 12

2.2.1 Gathering Requirements 12

2.2.2 System Analysis..... 12

2.2.3 System Design 12

2.2.4 Implementation 12

2.2.5 Project Schedule Gantt Chart 12

2.3 Feasibility Study 14

2.3.1 Focus 14

3 System Analysis 15

3.1 Overview..... 15

3.2 Framework15

3.3 System Analysis 16

3.3.1 Process Modeling 16

3.3.2 Requirements..... 17

3.3.3 Use Case Diagrams..... 18

3.3.4 Use Case Scenario 19

3.3.5 Activity Diagram 32

3.3.6 Non Functional Requirements 35

3.4	System Design.....	36
3.4.1	Sequence Diagram.....	36
3.4.2	Class Diagram	40
3.4.3	Relational Database Diagram (ERD)	41
4	System Development	42
4.1	Overview	42
4.2	App's architecture & design patterns	42
4.3	Used Technologies	42
4.3.1	Dart	42
4.3.2	Flutter	43
4.3.3	GetX	43
4.3.4	Google Cloud Platform	43
4.4	Mobile development	43
5	Drug-Drug Interaction and Drugs Recommendation	50
5.1	Experimental Results and Comparative Analysis to Drug-Drug Interaction Model	50
5.1.1	Experiment Specifications and Used Materials	50
5.1.2	Evaluation Metrics	50
5.1.3	Results of The System	51
5.2	Experimental Results and Comparative Analysis to Medicine Recommendation System Model	52
5.2.1	Experiment Specifications and Used Materials	52
5.2.2	Evaluation Metrics	52
5.2.3	Results of The System	54
6	Embedded System	56
6.1	Overview	56
6.2	Hardware components	56
6.2.1	ESP32	56
6.2.2	Servo Motor	58

6.2.3 Buzzer	58
6.2.4 LEDs	59
6.2.5 Box	60
7 Conclusions & Future Work	63
7.1 Conclusions	63
7.2 Future work	63

List of Figures

2.1 Gantt Chart	13
2.2 Project Schedule	13
3.1 Frame work	15
3.2 Context Diagram	16
3.3 DFD	17
3.4 Use Case	18
3.5 Patient Activity Diagram	32
3.6 Guardian Activity Diagram	33
3.7 Admin Activity Diagram	34
3.8 System Sequence Diagram	36
3.9 Login Sequence Diagram	37
3.10 Insert Drug Information Sequence Diagram	37
3.11 Receive DDI Warning Sequence Diagram	38
3.12 Receive Reminders Sequence Diagram	38
3.13 Take Drug Sequence Diagram	39
3.14 Follow Up Drugs' Schedule Sequence Diagram	39
3.15 Class Diagram	40
3.16 ERD	41
4.1 Splash Screen	44
4.2 Onboarding Screen 1	44
4.3 Onboarding Screen	44
4.4 Sign Up	45
4.5 Login	45
4.6 Reset Password	46
4.7 Account Screen	46
4.8 Edit Profile	46

4.8 Select Meal Time	46
4.9 Add New Medicine	47
4.10 Select Types	48
4.11 Active medications	48
4.12 Inactive medications	48
4.13 Drug Drug Interaction	49
4.14 Drug Drug No Interaction	49
4.15 Medications Alternatives	49
5.1 Drug Interaction Network	51
5.2 Dataset DDI	51
5.3 Dataset Medicine Recommendation	52
5.4 Dataset After Preprocessing	53
5.5 Vectors To Dataset Descriptions (tags)	53
5.6 Similarity Matrix To Medicine	54
5.7 Similarity Score To Recommended Medicines	55
6.1 ESP32	57
6.2 Servo Motor	58
6.3 Buzzer	59
6.4 LEDs	59
6.5 Box interface	60
6.6 Box Slots & Upper Doors	61
6.7 Box Drawer	62

List of Tables

3.1: Login Use Case Scenario	19
3.2: Patient Receive Reminders Use Case Scenario	20
3.3: Patient Take Drugs Use Case Scenario	21
3.4: Guardian Insert Drug Name Use Case Scenario	22
3.5: Guardian Receive DDI Warnings Use Case Scenario	23
3.6: Guardian Follow Up Drugs' Schedules Use Case Scenario	24
3.7: Guardian Search For Alternative Drugs Use Case Scenario	25
3.8: Admin Check DDI Use Case Scenario	26
3.9: Admin Send DDI Warning Use Case Scenario	27
3.10: Admin Send Reminders Use Case Scenario	28
3.11: Admin Send Missed Drug Use Case Scenario	29
3.12: Admin Check Alternative Drugs Use Case Scenario	30
3.13: Admin Send Alternative Drugs Use Case Scenario	31

Chapter 1

Introduction

1.1 Overview

In the last couple of decades, biomedical and smart healthcare research has witnessed a rapid amount of growth in terms of its presence in the literature, novel discoveries, and computational approaches, due to which a huge amount of experimental data has been generated and published (i.e., Big Data) to validate and to describe these innovations. The amount of data is so large that it is impossible for a human being to analyze and read the articles related to their desired field, and even the simple extraction of field-related published articles has become nearly impossible. As such, to utilize the information from scientific publications and articles, the increased use of automation due to the diversification and explosive growth in the healthcare literature as well as in the pharmaceutical industry is a clear representation of this growth. This is why attention has now been diverted and focused on the implementation and evolution of tools for knowledge-based discovery: the tremendous amount of biomedical research in the literature requires automation to extract, represent, interpret, and maintain this information in a refined manner. On the other hand, pills schedule management can be hard for certain types of patients taking into consideration their medical condition, age, or even their lifestyle. Therefore, we aim to help those who need pills, doctor appointments, and diet management and reminder. A dedicated mobile application with notifications as well as a physical box that allows patients with Dementia, Alzheimer, or any similar condition to remember their medicine at the right time in the right order.

1.2 Problem definition

Medication management can be difficult for patients, especially when it comes to remembering which medications to take at what time and taking drugs that they do not know about their interaction with other drugs they also taking. This can become more complicated if the patient has to control and differentiate between multiple medications. The caregiver needs to be with the patient all the time, give him medicine on time and make sure that the patient is fine, which is so stressful and with the daily routine, it becomes so hard.

1.3 Project Motivation

We believe that life is about helping others and giving back to our society!

Imagine a world where every patient feels cared for, supported, and safe while managing their medications. Our innovative medication care system brings this vision to life, ensuring patients never feel alone in their health journey. By incorporating a dedicated caregiver who supervise their drugs, patients receive timely reminders for each dose, transforming medication management from a hard task into a seamless part of their daily routine. This personal touch not only ensures adherence but also offers emotional support, easing the hardness of managing chronic conditions. The system's intelligent drug interaction checker acts as a guardian angel, which protect the patient from potential adverse effects by alerting the caregiver to any harmful combinations. This heartfelt approach to medication care fosters a deep sense of security and trust, allowing patients to focus on living their lives to the fullest. Our medication care system means embracing a future where health and human connection are intertwined, providing comfort and reassurance at every step.

1.4 Project Objectives

- Medication Management: An application that helps store and track all patients' medications, especially the elderly and those who have a strict medication regimen.
- Reminders: Send reminders to patients about when to take their medications.
- Recommendation Drugs: Giving alternative drugs
- Patient Supervision: An application that allows the person responsible for the patient to oversee the patient's healthcare.
- Avoid drugs interactions.

1.5 Organization

We have structured the rest of this project as follows: Chapter (2) presents the planning Phase is the fundamental process of understanding why a system should be built and also determining how the project team will go about building the system, Chapter (3) presents project analysis and reviews the Development methodology, the Agile Software Development Life Cycle also reviews the functional and non-functional requirements, Chapter(4)presents experimental results and comparative analysis to Drug-Drug Interaction (DDI) Model and alternative drugs recommendation system model, data Source, Libraries and Tools Used and results Chapter(5)presents hardware components used.

Chapter 2

Project Planning

2.1 Introduction

In This chapter will discuss the development methodology used in the project and how it was accomplished, the plan that our team follow to implement the whole project and, the team members and their role and tasks they done.

2.2 Project Plan

Project planning phase is a discipline addressing how to complete a project in a certain timeframe, usually with defined stages and designated resources.

2.2.1 Gathering Requirements

The process of identifying, understanding, and documenting the needs and expectations of stakeholders for a project or system to ensure the final result meets business objectives and user needs.

2.2.2 System Analysis

- Functional and non-functional requirements
- DFD (Data flow diagram)
- Use Case

2.2.3 System Design

- Sequence diagram
- Class diagram
- Activity diagram
- Context diagram

2.2.4 Implementation

- Mobile Application
- ML model
- Embedded Systems

2.2.5 Project Schedule Gantt Chart

Is a visual representation of a project's tasks, durations, and dependencies over time. It displays tasks as horizontal bars along a timeline, indicating when each task starts, ends, and its relationships with other tasks.

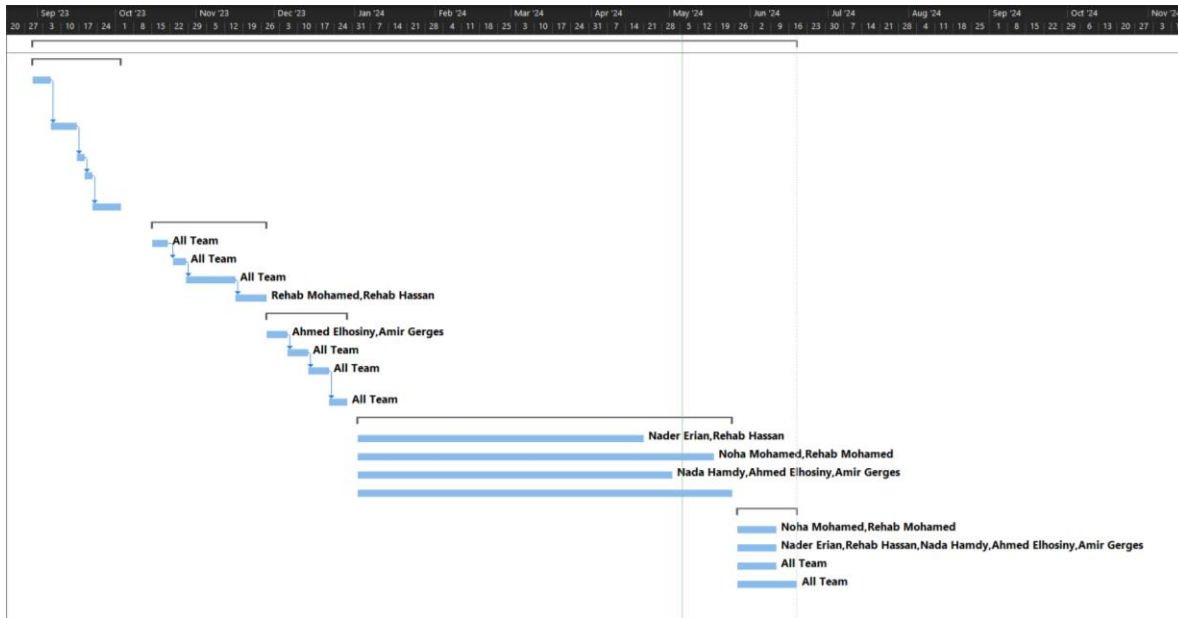


Figure 2.1: Gantt Chart

Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
	تخطيط	215 days?	Wed 8/30/23	Tue 6/18/24		
	Planning	25 days?	Wed 8/30/23	Mon 10/2/23		All Team
	Collect Advantages and Disadvantages from related works	5 days?	Wed 8/30/23	Tue 9/5/23		
	Check data availability	8 days	Wed 9/6/23	Fri 9/15/23	3	
	Define project tasks	2 days?	Sat 9/16/23	Mon 9/18/23	4	
	Assign tasks to team members	3 days?	Tue 9/19/23	Thu 9/21/23	5	
	Build Block Diagram	7 days	Fri 9/22/23	Mon 10/2/23	6	
	System Analysis	34 days?	Sun 10/15/23	Mon 11/27/23		
	Context Diagram	6 days?	Sun 10/15/23	Fri 10/20/23		All Team
	DFD	5 days	Mon 10/23/23	Fri 10/27/23	9	All Team
	Use Case	14 days	Sat 10/28/23	Wed 11/15/23	10	All Team
	ERD	8 days?	Thu 11/16/23	Mon 11/27/23	11	Rehab Mohamed, Rehab Hassan
	System Design	23 days?	Tue 11/28/23	Thu 12/28/23		
	Class Diagram	6 days?	Tue 11/28/23	Tue 12/5/23		Ahmed Elhosiny, Amir Gerges
	Activity Diagram	6 days	Wed 12/6/23	Wed 12/13/23	14	All Team
	System Sequence Diagram	6 days?	Thu 12/14/23	Thu 12/21/23	15	All Team
	Sequence Diagram	5 days?	Fri 12/22/23	Thu 12/28/23	16	All Team
	Implementation	105 days?	Tue 1/2/24	Fri 5/24/24		
	Mobile Application	80 days?	Tue 1/2/24	Sat 4/20/24		Nader Erian, Rehab Hassan
	Internet Of Things	100 days?	Tue 1/2/24	Fri 5/17/24		Noha Mohamed, Rehab Mohamed
	Machine Learning	88 days	Tue 1/2/24	Wed 5/1/24		Nada Hamdy, Ahmed Elhosiny, Amir Gerges
	Back End	105 days?	Tue 1/2/24	Fri 5/24/24		
	Testing	17 days?	Mon 5/27/24	Tue 6/18/24		
	Components Testing	11 days?	Mon 5/27/24	Mon 6/10/24		Noha Mohamed, Rehab Mohamed
	Application Testing	11 days?	Mon 5/27/24	Mon 6/10/24		Nader Erian, Rehab Hassan, Nada Hamdy, Ahmed Elhosiny, Amir Gerges
	Back End Testing	11 days	Mon 5/27/24	Mon 6/10/24		All Team
	Documentation	17 days?	Mon 5/27/24	Tue 6/18/24		All Team

Figure 2.2: Project Schedule

2.3 Feasibility Study

The feasibility study outlines and analyzes several alternatives or methods of achieving business success.

2.3.1 Focus

Helping patients with Dementia, Alzheimer, or any similar condition to remember their medicine at the right time in the right order.

Chapter 3

System Analysis

3.1 Overview

In the lifecycle of software development, system analysis and design are crucial stages. These phases involve understanding the requirements, conceptualizing the system architecture, and creating detailed designs to guide the development process. This chapter covers the essential aspects of system analysis and design, highlighting their significance in ensuring the successful implementation of documentation projects.

3.2 Framework

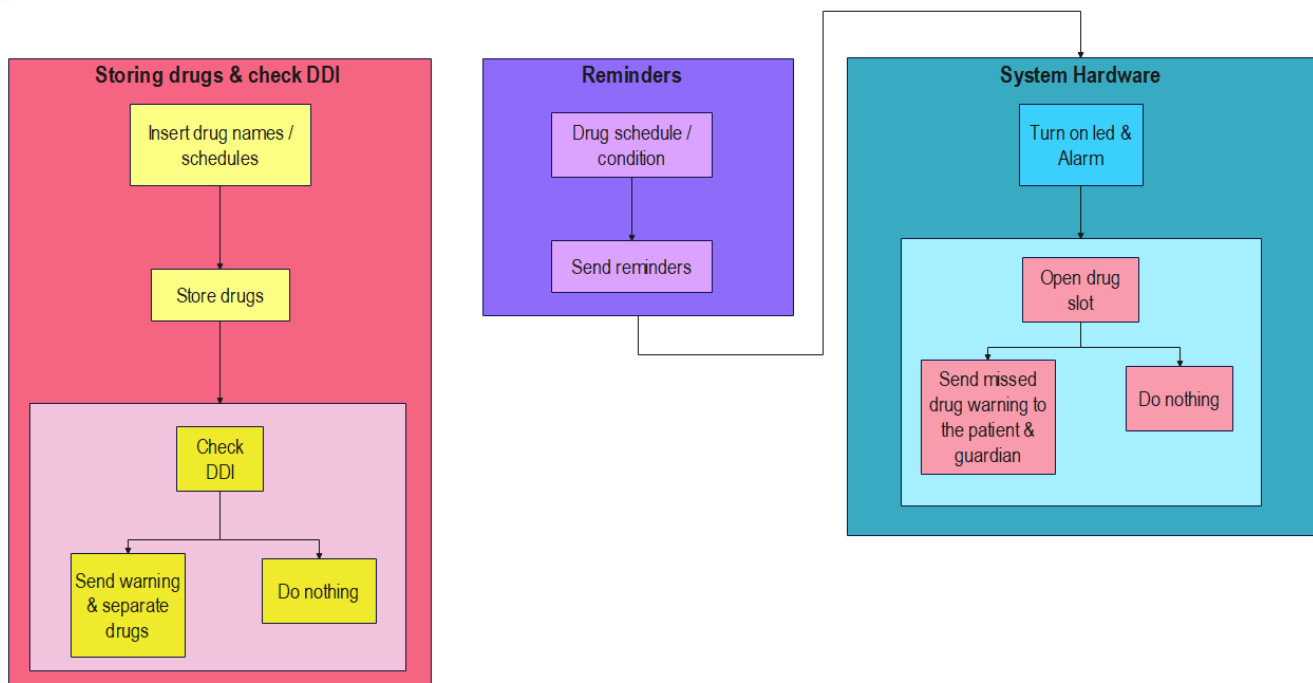


Figure 3.1: Framework

3.3 System Analysis

System analysis is a critical phase in the software development lifecycle where the primary goal is to understand and specify what a system should do to meet the needs of its users and stakeholders. It involves a comprehensive examination of the existing system, the identification of system requirements, and the establishment of a detailed specification that serves as a foundation for system design and development.

3.3.1 Process Modeling

Context Diagram

A context diagram is a high-level, simplified representation of a system that shows the system as a single process along with its interactions with external entities. It provides an overview of the system boundaries and the external agents that interact with it, making it easier to understand the scope and context of the system.

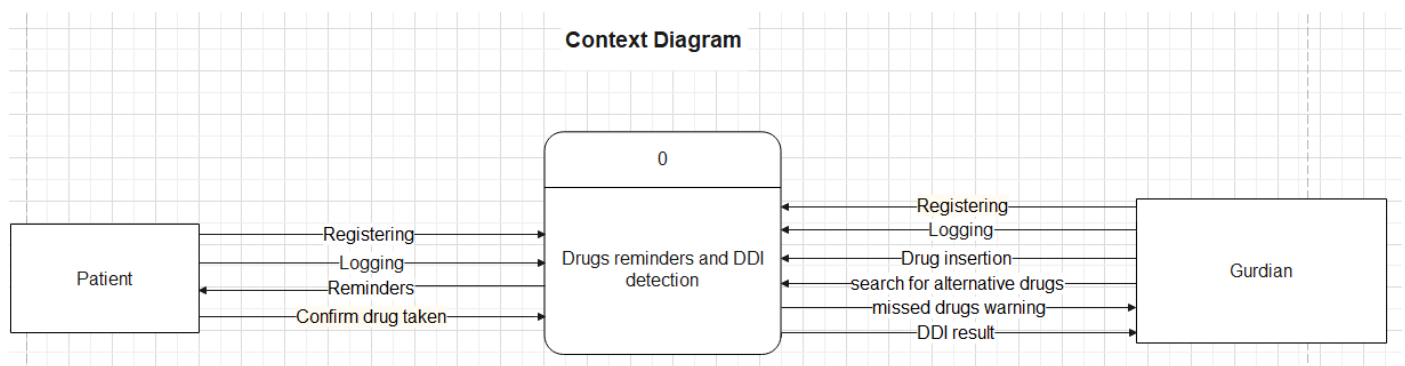


Figure 3.2: Context Diagram

Data Flow Diagram

A DFD illustrates how data is processed by a system in terms of inputs and outputs. It includes processes, data stores, data flows, and external entities.

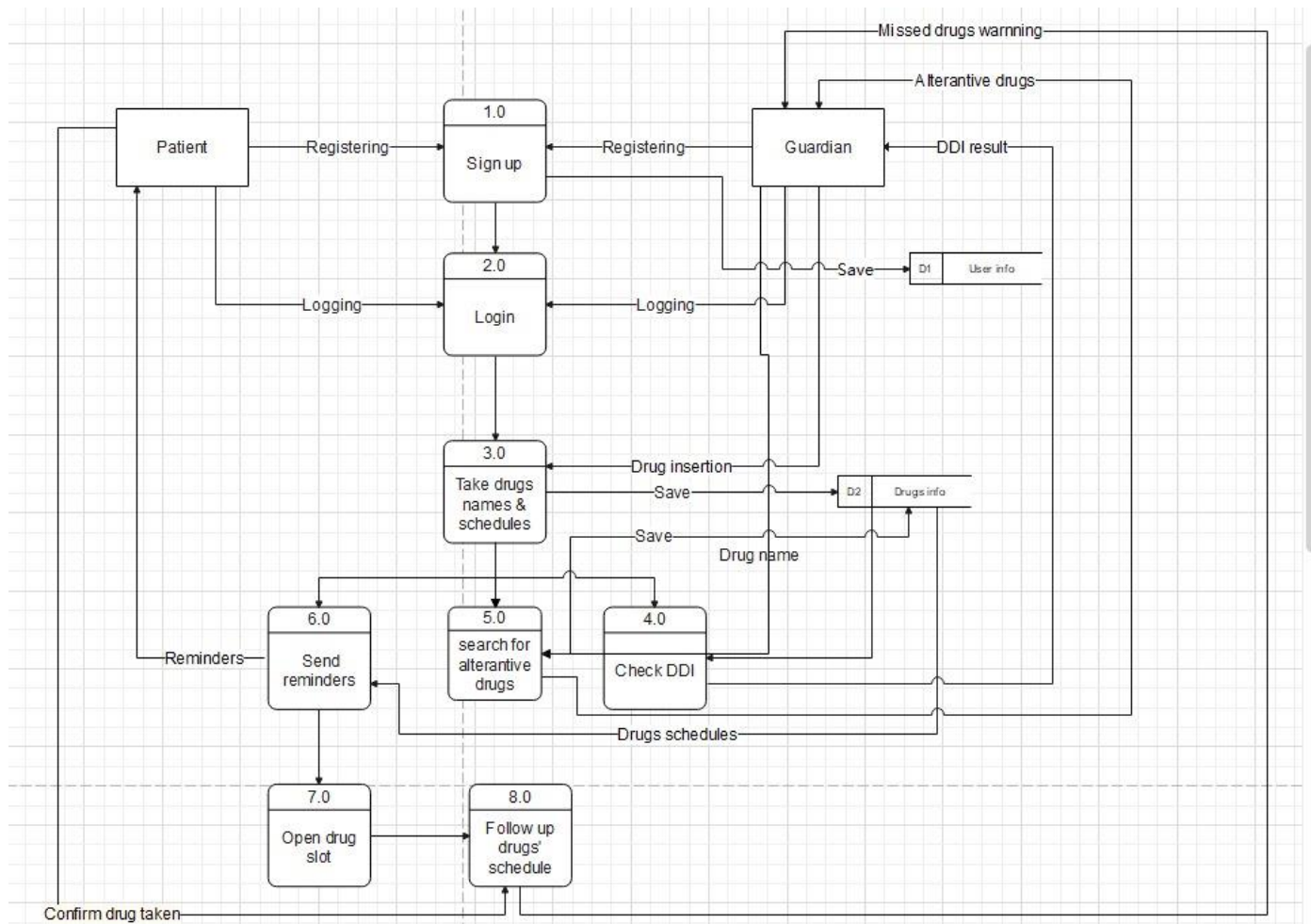


Figure 3.3: DFD

3.3.2 Requirements

Functional Requirements

Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work. Functional requirements are product features and focus on user requirements. Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met; the system will not work. Functional requirements are product features and focus on user requirements.

3.3.3 Use Case Diagrams

Use Case Diagrams are a type of Unified Modeling Language (UML) diagram that visually represents the functional requirements of a system. They illustrate the interactions between users (actors) and the system, capturing the different ways the system will be used. Use case diagrams help in understanding the system’s functionality from an end-user perspective and are useful for communicating the system’s requirements to stakeholders.



Figure 3.4: Use Case

3.3.4 Use Case Scenario

	Use case Name:		Login
	Actor(s):		Guardian & Patient
+	Description:		This is a mandatory use case the user is going to interact with in the application and he has to be logged in order to use the app. The user is required to inter his/her correct account as asked from them.
	Typical Course of Events:	Actor Action	System response
		Step 1: The user wants to enter use their account. Step 3: The user enters the required fields.	Step 2: The system asks the user to enter their correct email and password. Step 4: System checks correctness of the given account. Step 5: System expresses its state so the user is required to enter a valid account or they're logged in successfully.
	Alternate Courses:	Step3: If the user forgot their password, they're going to redeem making a new password. Step 4: if information isn't correct, go to step 2 and repeat.	
	Precondition:	Signup.	
	Post condition:	Nothing.	
	Assumption:	If the user is already logged in, they're supposed to use the app without this step.	

Table 3.1: Login Use Case Scenario

Patient Use Cases

Use case Name:	Receive reminders	
Actor(s):	Patient	
Description:	The use case describes the process where the patient receives the reminders.	
Typical Course of Events:	Actor Action	System Response
	Step 1: After the guardian inserts patient's drugs information.	Step 2: System sends a reminder notification at the set time. Step 3: System marks the reminder as complete.
Alternate Courses:	Step 3: System may send a reminder multiple times if the patient doesn't acknowledge it.	
Precondition:	The guardian must have set up a drug schedule.	
Post condition:	The patient has acknowledged the reminder.	
Assumption:	None.	

Table 3.2: Patient Receive Reminders Use Case Scenario

Use case Name:	Take drugs	
Actor(s):	Patient	
Description:	This use case describes a process in which a patient takes their drugs after receiving a reminder notification at the specified time and opens the drug box at the same time as receiving the notification.	
Typical Course of Events:	Actor Action	System Response
	Step 3: After the patient receives notification, the patient will take the drug. Step 4: The patient confirms on the Drug Management System that he has taken the drugs.	Step 1: The system sends a notification at the specified time. Step 2: The system sends an opening command to the box. Step 5: After receiving confirmation from the patient that he has taken the drugs, the system will lock the box.
Alternate Courses:	Step 3: if the pill falls. The patient must communicate the guardian.	
Precondition:	The system sends notifications at the specified time.	
Post condition:	The system locks the box.	
Assumption:	None.	

Table 3.3: Patient Take Drugs Use Case Scenario

Guardian Use Cases


		
Use case Name:	Insert drugs names & schedules	
Actor(s):	Guardian	
Description:	This use case describes the process done by the guardian to insert patient's drugs names & schedules to the system.	
Typical Course of Events:	Actor Action	System response
	Step1: After login to the system the guardian wants to insert the drugs names & schedules to deal with it. Step2: The guardian inserts patient's drugs names, schedules and all required information.	Step3: The system stores drugs' information. Step4: The system sends notifications at the specified time.
Alternate Courses:	Step2: if the drug is already existing, the system sends notification.	
Precondition:	Login to the system.	
Post condition:	- Drugs information has been recorded. - The system sends notifications at the specified time.	
Assumption:	None.	

Table 3.4: Guardian Insert Drug Name Use Case Scenario


		
Use case Name:	Receive DDI Warnings	
Actor(s):	Guardian	
Description:	The use case describes that the guardian receives notification if there is any drug-drug interaction (DDI).	
Typical Course of Events:	Actor Action	System Response
	Step1: This use case is initiated when the guardian inserts drug names. Step4: The guardian receives this warning.	Step2: The system checks if there is any drugs interaction. Step3: The system sends DDI warning to guardian if there is an interaction.
Alternate Courses:	Nothing.	
Precondition:	System checks for any DDI.	
Post condition:	The guardian acknowledges the warning.	
Assumption:	If there isn't any interaction, the system doesn't send notification to guardian.	

Table 3.5: Guardian Receive DDI Warnings Use Case Scenario



Use case Name:	Follow up drugs' schedules	
Actor(s):	Guardian	
Description:	The use case describes that Guardian tracks whether the patient has taken drugs or not.	
Typical Course of Events:	Actor Action	System Response
	Step1: This use case is initiated when the guardian inserts the schedules of the drugs.	Step2: Check if the patient has taken drugs or not. Step3: If the patient didn't take the drugs, the system sends notification to guardian
Alternate Courses:	Nothing	
Precondition:	The guardian inserts drugs schedules.	
Post condition:	The system sends notification to guardian.	
Assumption:	If the patient took the drugs, the system doesn't send notification to guardian.	



Table 3.6: Guardian Follow Up Drugs' Schedules Use Case Scenario

Use case Name:	Search for alternative drugs	
Actor(s):	Guardian	
Description:	The use case describes the process where the guardian search for alternative drugs.	
Typical Course of Events:	Actor Action	System Response
	Step 1: This use case is initiated when the Guardian enter drugs name to search for alternative them.	Step 2: The system shows the alternative drugs.
Alternate Courses:	Step 2: The system may send a specific message if alternative drugs aren't existed.	
Precondition:	Login to the system	
Post condition:	The guardian acknowledges the alternative drugs.	
Assumption:	None.	

Table 3.7: Guardian Search For Alternative Drugs Use Case Scenario

Admin Use Cases

Use case Name:	Check DDI	
Actor(s):	Admin	
Description:	The use case describes the process done by the system to check if there is any drug-drug interaction.	
Typical Course of Events:	Actor Action	System Response
	Step1: This use case is initiated when the guardian inserts drugs names. Step 2: System checks for any DDI. Step 3: If there is any DDI, the system separates this drug.	
Alternate Courses:	Nothing.	
Precondition:	Guardian inserts drugs names.	
Post condition:	The system sends DDI warning to the guardian.	
Assumption:	If there isn't any interaction, the system doesn't send notification to guardian.	

Table 3.8: Admin Check DDI Use Case Scenario

Use case Name:	Send DDI warning	
Actor(s):	Admin	
Description:	The use case describes the system when sending the result of DDI detection.	
Typical Course of Events:	Actor Action	System Response
	Step1: This use case is initiated when the system checks DDI. Step 2: If there is a DDI, it sends warning to the guardian.	
Alternate Courses:	Nothing.	
Precondition:	System Checks for DDI.	
Post condition:	The guardian receives the warning.	
Assumption:	If there isn't any interaction, the system doesn't send notification to guardian.	

Table 3.9: Admin Send DDI Warning Use Case Scenario

Use case Name:	Send reminders	
Actor(s):	Admin	
Description:	The use case describes the system when sending the reminders based on drugs conditions.	
Typical Course of Events:	Actor Action	System Response
	Step1: This use case is initiated when guardian inserts drugs information. Step 2: The system sends the reminders to the patient based on this information.	
Alternate Courses:	Nothing.	
Precondition:	The Guardian inserts drugs information.	
Post condition:	The system checks if there are any missed drugs.	
Assumption:	If the patient doesn't acknowledge the reminders the system sends warning to the patient and the guardian.	

Table 3.10: Admin Send Reminders Use Case Scenario

Use case Name:	Send missed drug warning	
Actor(s):	Admin	
Description:	The use case describes the system when sending warning to the patient and the guardian if the patient doesn't take the drug at specific time.	
Typical Course of Events:	Actor Action	System Response
	<p>Step1: This use case is initiated when guardian insets drugs information.</p> <p>Step 2: The system opens drug slot at drug's time.</p> <p>Step 3: The system checks if the patient takes the drug or not.</p> <p>Step 4: If the patient doesn't take the drug it sends warning to both of them.</p>	
Alternate Courses:	Nothing.	
Precondition:	The guardian inserts drugs information.	
Post condition:	Acknowledge the warning.	
Assumption:	None.	

Table 3.11: Admin Send Missed Drug Use Case Scenario

Use case Name:	Check alternative drugs	
Actor(s):	admin	
Description:	The use case describes the process where the admin checks alternative drugs.	
Typical Course of Events:	Actor Action	System Response
	<p>Step 1: This use case is initiated when the Guardian enter drugs name.</p> <p>Step 2: The admin check if drug have alternative drugs existed or not.</p>	
Alternate Courses:		
Precondition:	The Guardian enter drug name	
Post condition:	The system sends the alternative drugs.	
Assumption:	None.	

Table 3.12: Admin Check Alternative Drugs Use Case Scenario

Use case Name:	Send alternative drugs	
Actor(s):	Admin	
Description:	The use case describes the process where the admin sends alternative drugs.	
Typical Course of Events:	Actor Action	System Response
	Step 1: This use case is initiated when the admin checks alternative drugs. Step 2: if drug have alternative drugs, it sends them	
Alternate Courses:	Step 2: if drug haven't alternative drugs, it sends specific message.	
Precondition:	The admin checks alternative drugs.	
Post condition:	The guardian acknowledges the alternative drugs.	
Assumption:	None.	

Table 3.13: Admin Send Alternative Drugs Use Case Scenario

3.3.5 Activity Diagram

An activity diagram is a visual representation used in software engineering to illustrate the flow of actions or activities within a system or process, helping to understand, analyze, and communicate system behavior.

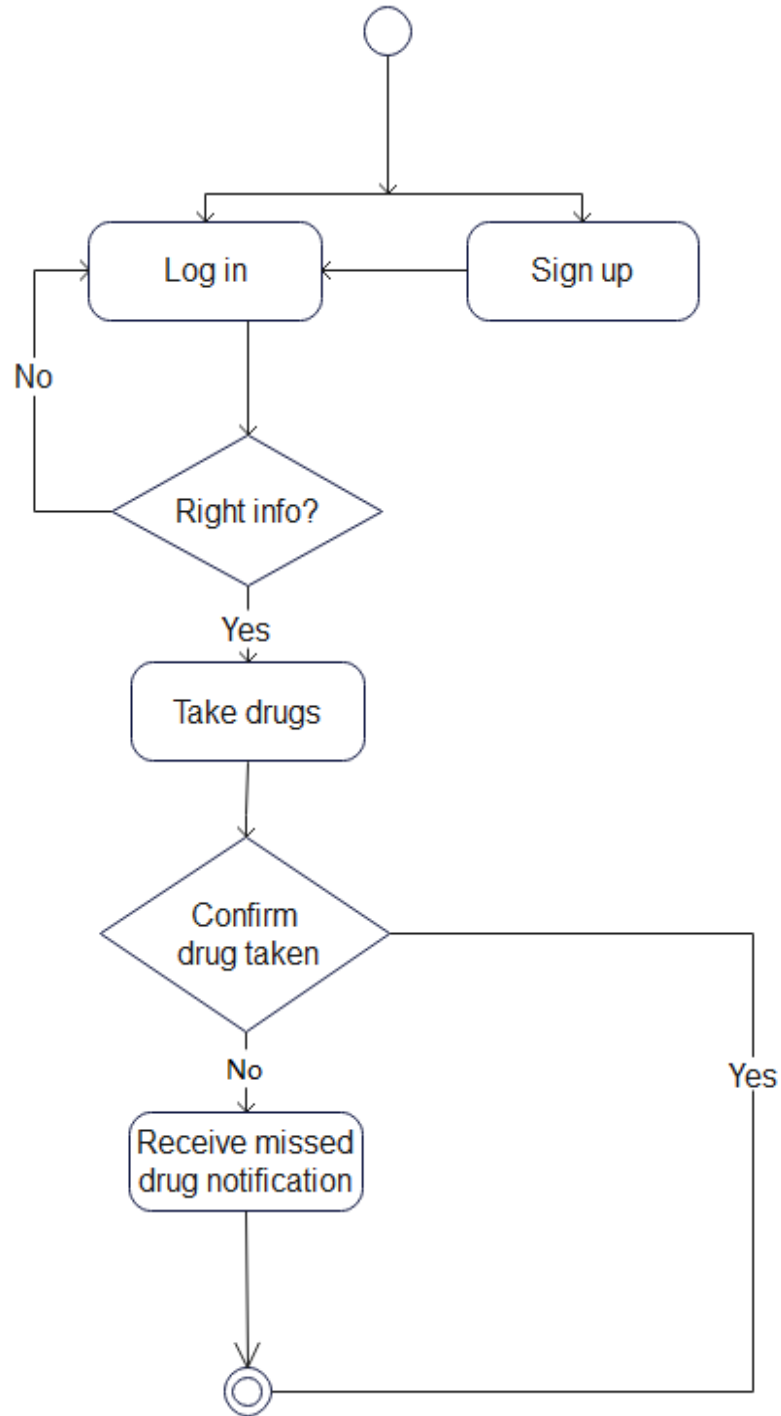


Figure 3.5: Patient Activity Diagram

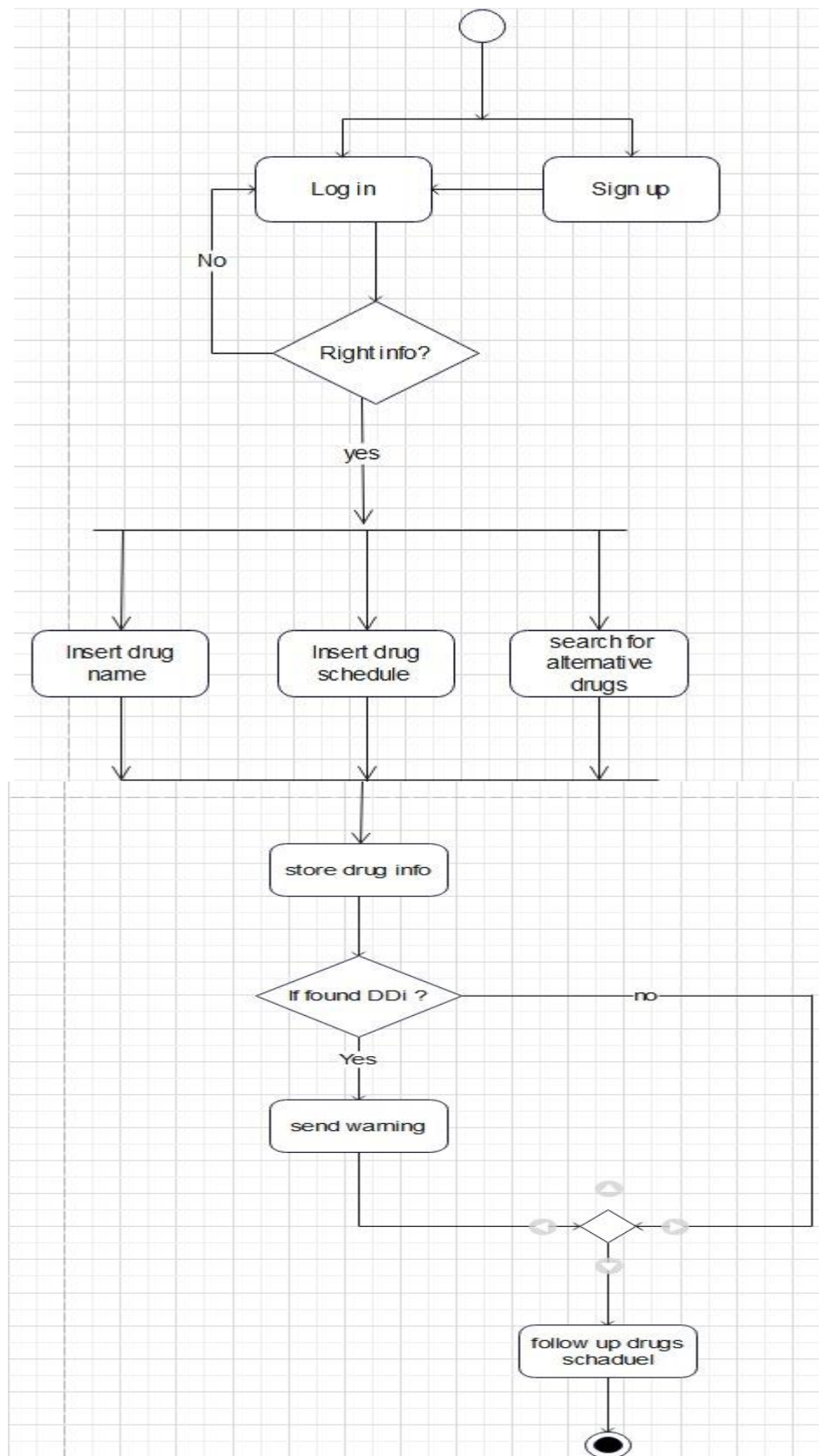


Figure 3.6: Guardian Activity Diagram

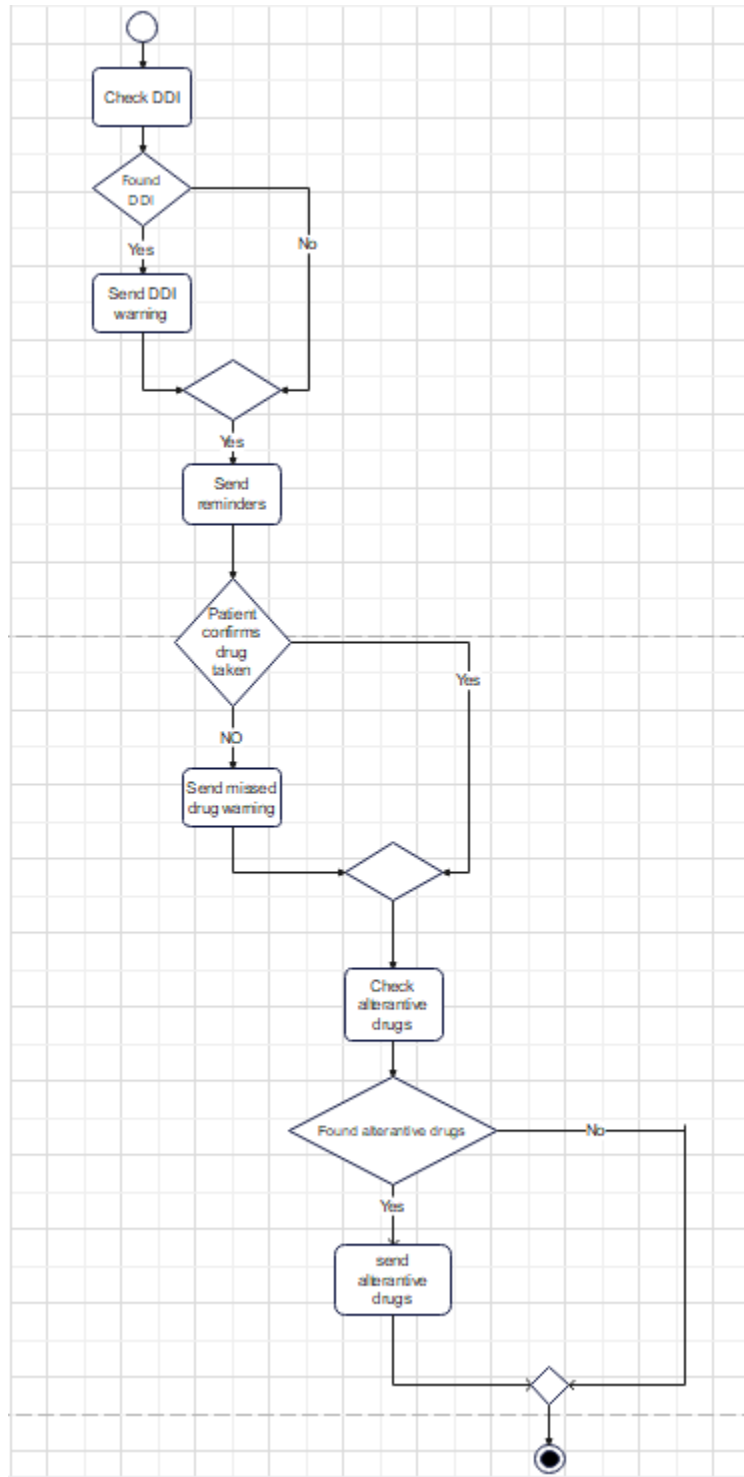


Figure 3.7: Admin Activity Diagram

3.3.6 Non-Functional Requirements

Non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behavior.

They are contrasted with functional requirements that define specific behavior or functions.

- Usability: The Mobile Application cart should be easily usable by the Patient as well as the care giver.
- Accessibility: the caregiver will receive the reminders in case of interactions, missed drug warning, on the other hand, the patient will receive drug reminders at the appropriate times.
- Performance: The main function must have specific real-time to be executed for each operation to not delay the whole system.
- Functions must be coded in the best ways following the design standards to produce well-behaved system.
- Security: The patient must be sure that all his details will be secured in the App options. Data shouldn't be sold under any name as it's confidential and in case of scientific research patient and care giver (guardian) must allow such thing.
- User-friendly: Easy to use and manipulate every proposed feature. This is done starting from app color harmony passing with user interface. Using easy to guess icons to make it easy for patients to use the mobile app.

3.4 System Design

The system design process refers to the series of steps or stages followed to create a detailed plan for the development of a system. While specific methodologies may vary depending on the context and requirements.

3.4.1 Sequence Diagram

A sequence diagram is a type of interaction diagram that shows how objects interact in a particular sequence to perform a specific task within a system. It illustrates the flow of messages between objects over time.

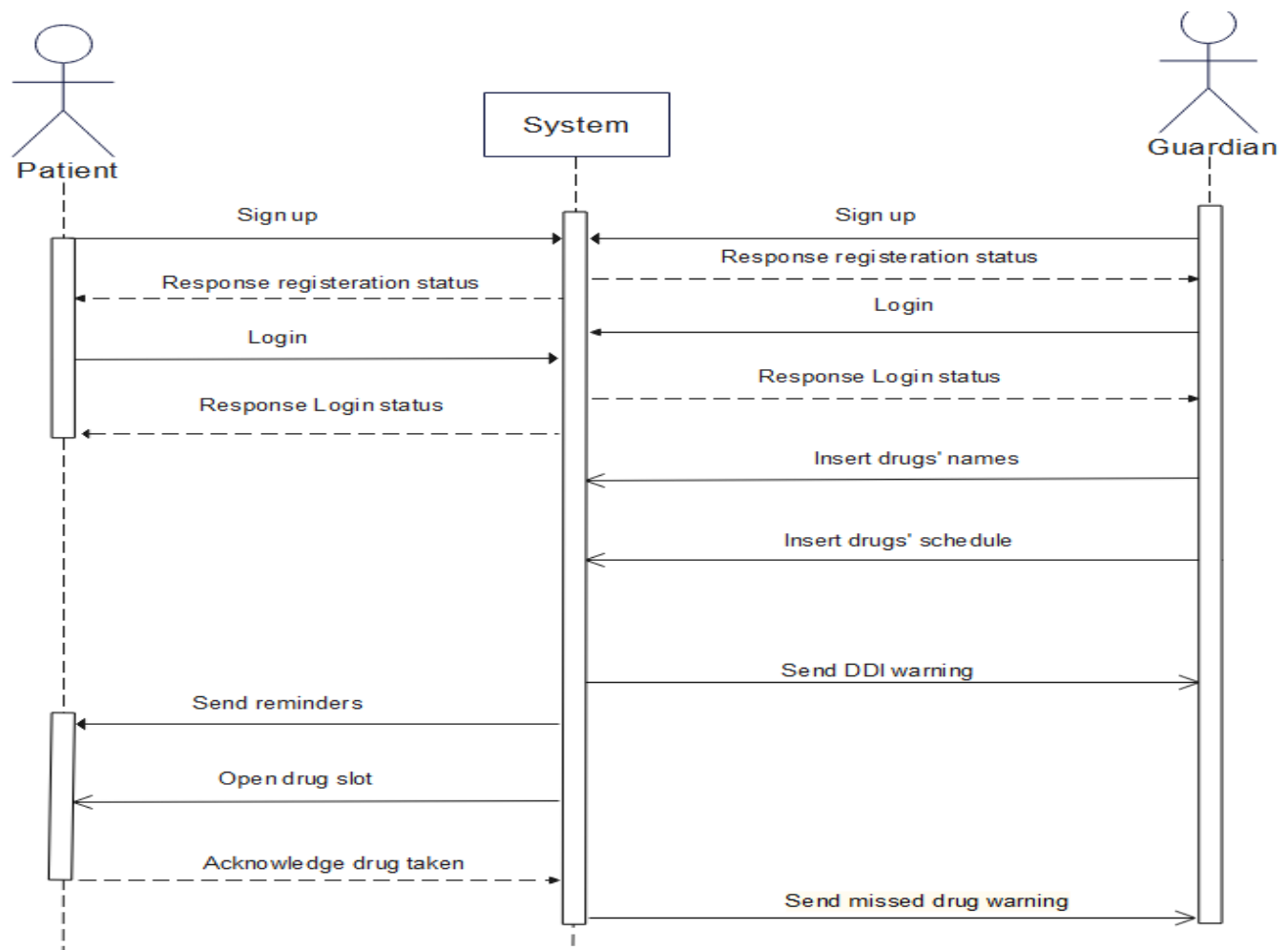


Figure 3.8: System Sequence Diagram

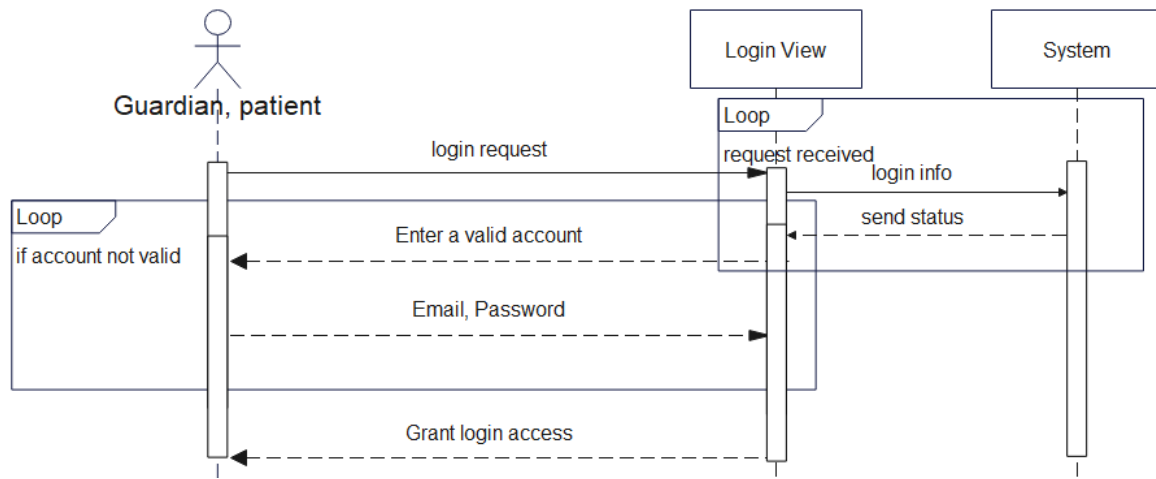


Figure 3.9: Login Sequence Diagram

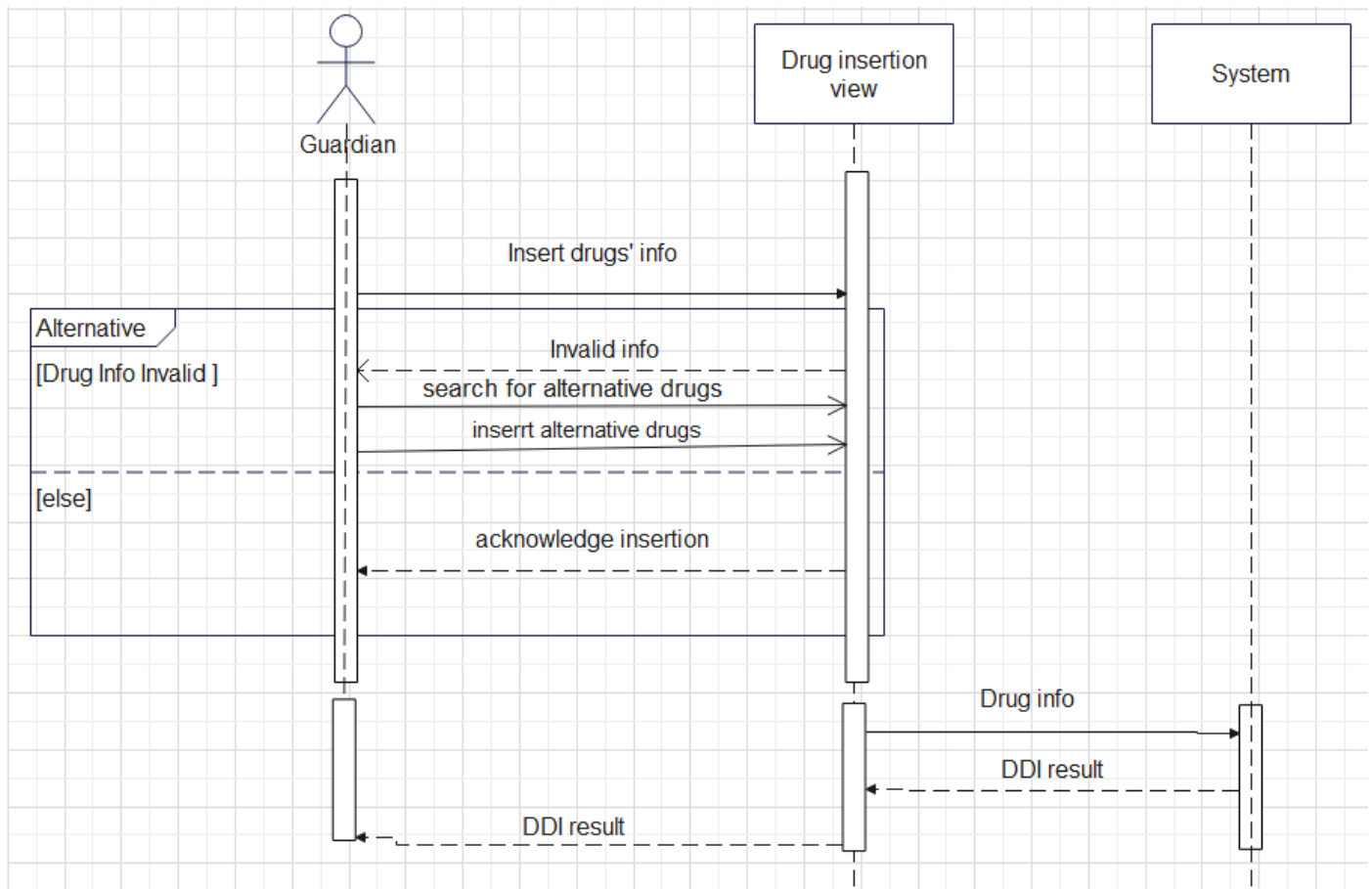


Figure 3.10: Insert Drug Information Sequence Diagram

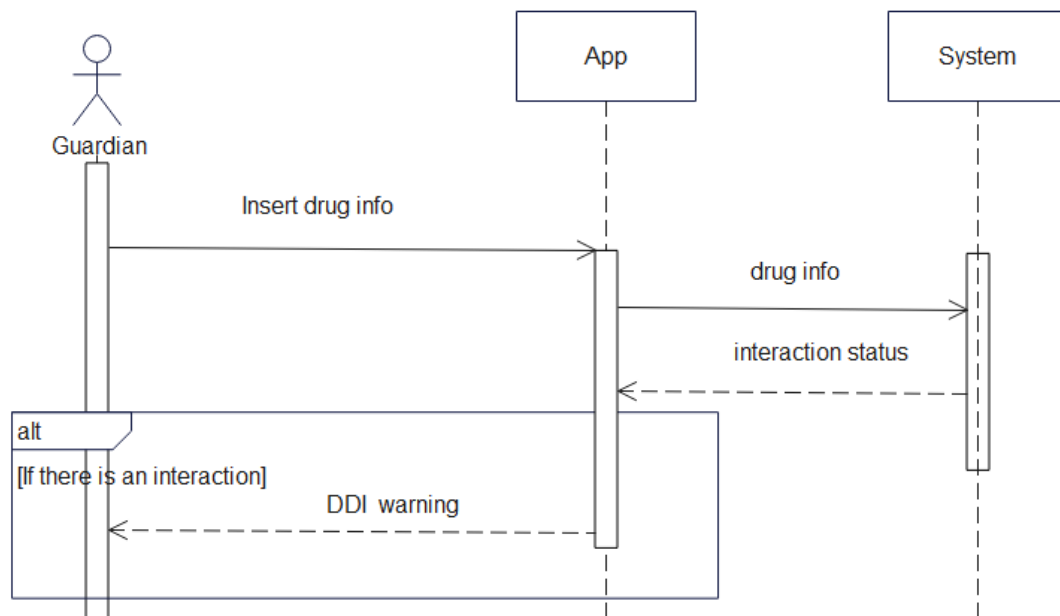


Figure 3.11: Receive DDI Warning Sequence Diagram

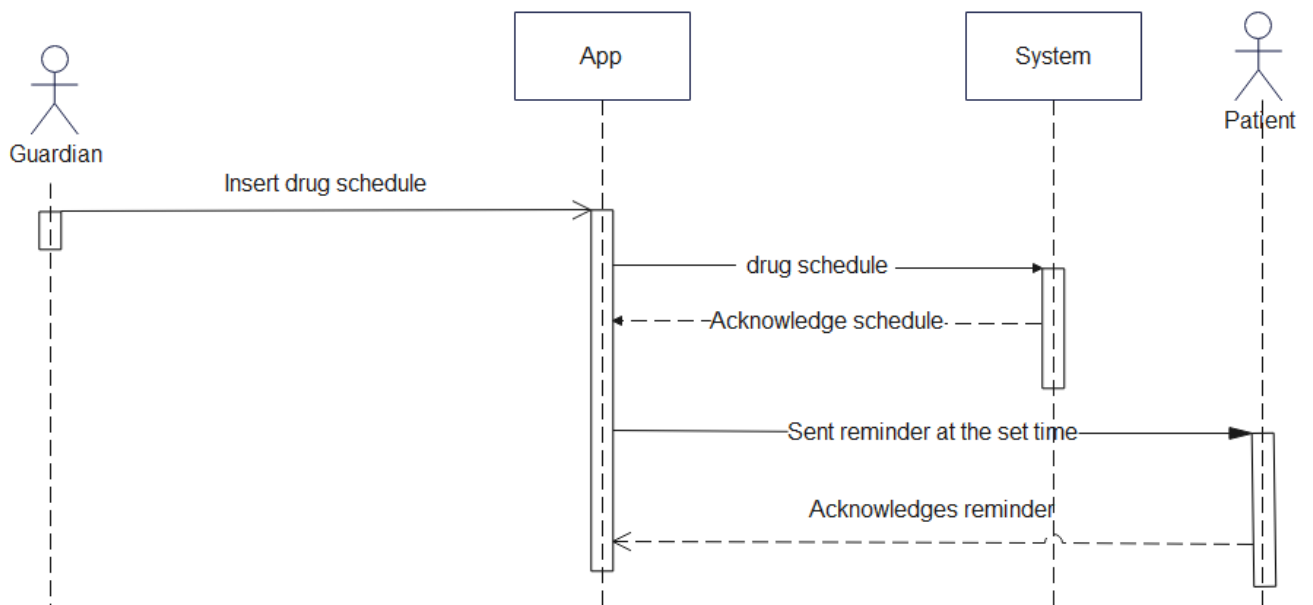


Figure 3.12: Receive Reminders Sequence Diagram

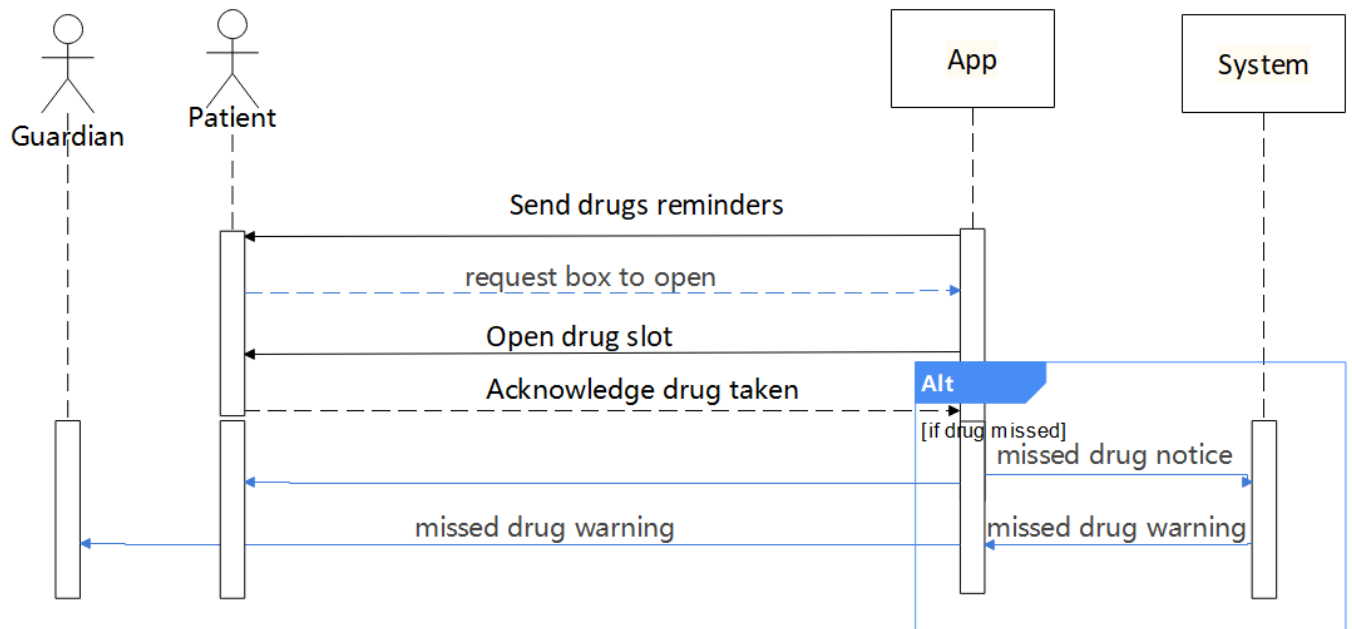


Figure 3.13: Take Drug Sequence Diagram

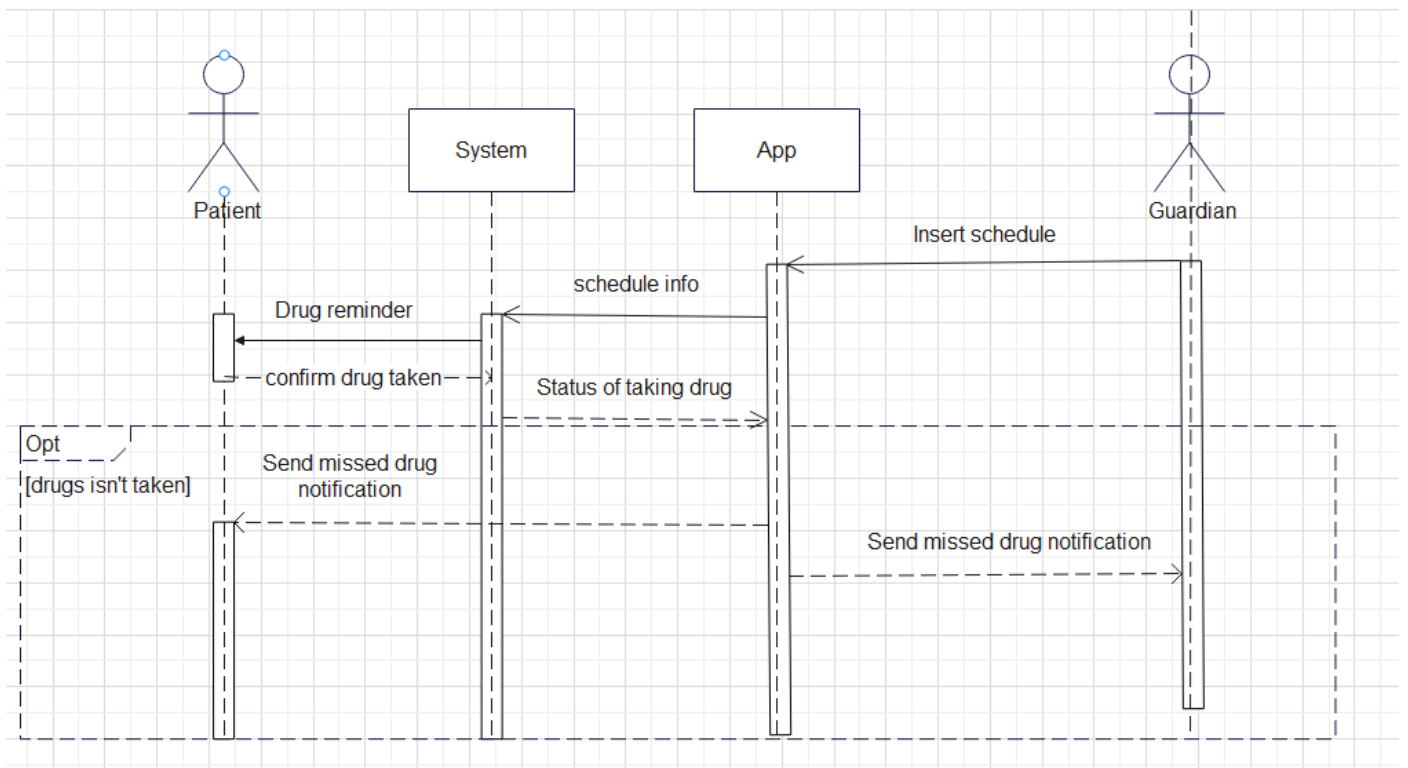


Figure 3.14: Follow Up Drugs' Schedule Sequence Diagram

3.4.2 Class Diagram

A class diagram is a type of static structure diagram in the Unified Modeling Language (UML) that represents the structure of a system by showing the classes of the system, their attributes, methods, relationships, and constraints.

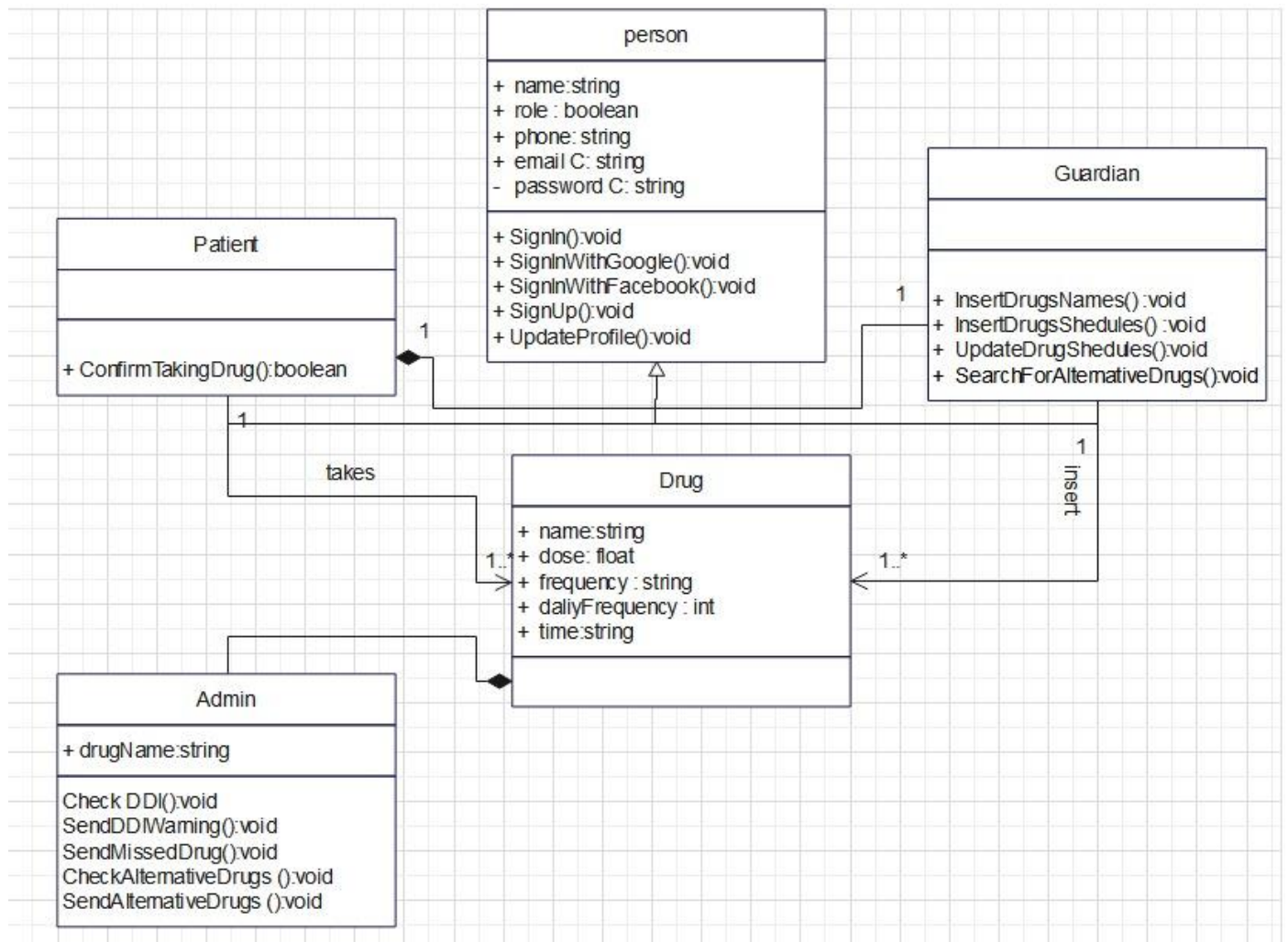


Figure 3.15: Class Diagram

3.4.3 Relational Database Diagram (ERD)

An Entity-Relationship Diagram (ERD) is a visual representation of entities (objects or concepts), attributes (properties), and relationships between entities within a database or information system. It helps in understanding the structure of the database, including how data is organized and how entities are related to each other.

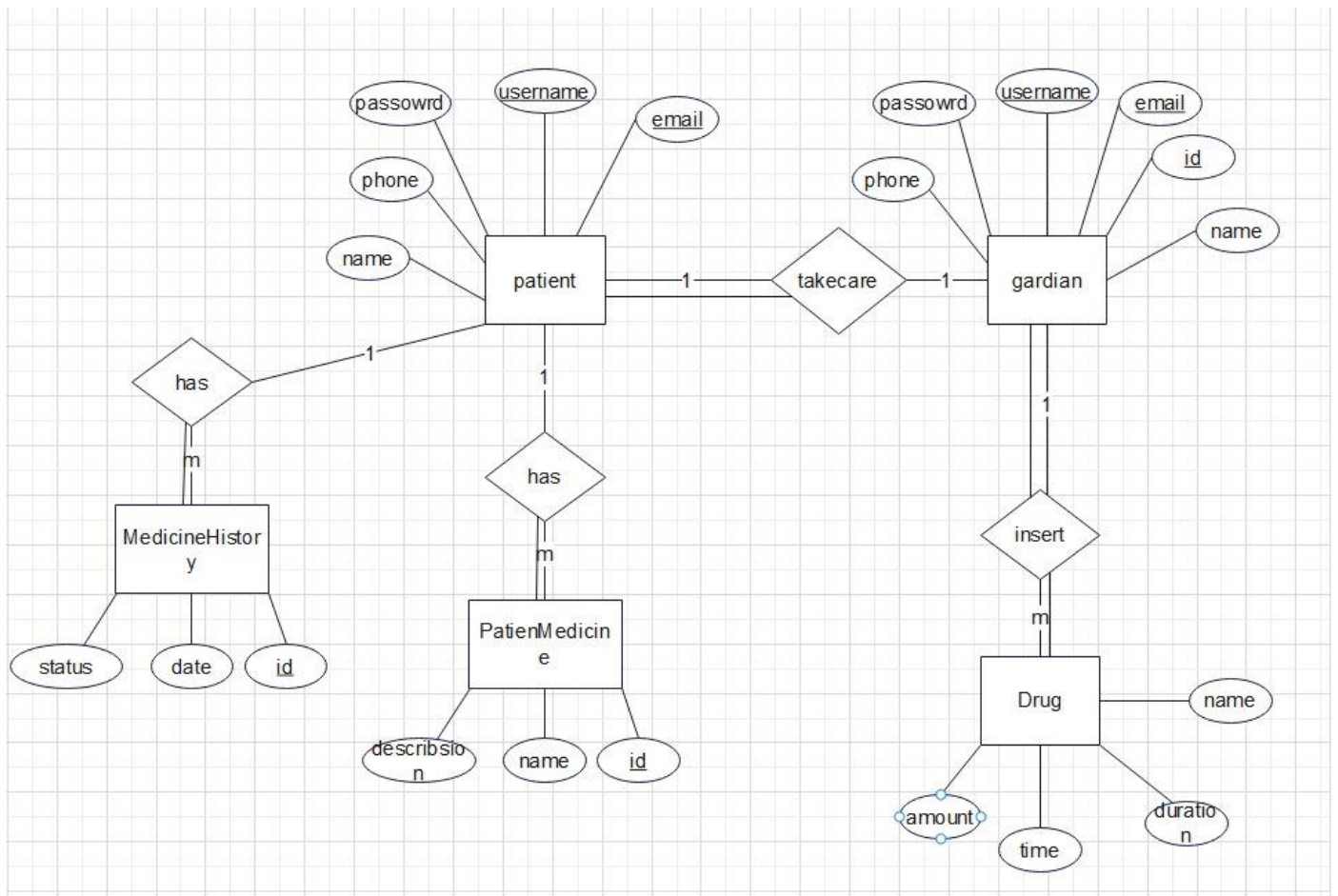


Figure 3.16: ERD

Chapter 4

System Development

4.1 Overview

Our application, developed using Flutter, is a comprehensive tool designed for streamline medication management, particularly for the elderly and those on strict medication schedules. It offers three distinct roles: Normal Patient, Caregiver, and Care Receiver. Normal Patients and Caregivers can add medications, while Care Receivers can view the medications added by their caregivers. A standout feature of our system is the medication box. This box stores the medication pills and releases them when it's time for the patient to take their medication. This ensures that the right medication is taken at the right time, further enhancing patient safety and adherence to medication regimens. When a Normal Patient or Caregiver adds a medication to the app, The patient then receives timely reminders from the app, and the medication is dispensed from the box at the appropriate time. Caregivers can add medications but do not receive notifications, allowing them to manage the medication list for their care receivers without any distractions. The application also integrates an AI model that checks for potential drug-drug interactions, ensuring patient safety. Moreover, it provides a feature to check for interactions between medications and suggests alternatives if necessary, ensuring that patients always have access to the safest and most effective treatment options.

4.2 App's architecture & design patterns

Our application employs a client-server architecture, with the client being the Flutter app. The foundation of the app is built on the Model-View-View Model (MVVM) design pattern. This pattern ensures a clear separation of concerns and enables effective code organization. This combination of architecture and design pattern guarantees that our application is scalable, maintainable, and easily extensible.

4.3 Used Technologies

4.3.1 Dart

Dart is a programming language optimized for building fast apps on any platform. It aims to be a productive language for multi-platform development, with a flexible execution runtime platform for app frameworks. Dart is versatile and well-suited for a range of development tasks, providing tools for building fast, reliable, and high-quality applications.

4.3.2 Flutter

Flutter is an open-source UI framework by Google for building cross- platform applications from a single codebase. It enables developers to cre ate responsive applications for Android, iOS, Windows, Mac, and Linux. Flutter’s native performance, hot reload feature, and comprehensive set of widgets and APIs make it a versatile and powerful tool for building high-quality applications with ease.

4.3.3 GetX

A lightweight and powerful state management solution for Flutter that combines high-performance state management, intelligent dependency injection, and route management. It helps to separate the UI from business logic while providing a clear and efficient way to manage routes. It uses fewer resources as possible and does not depend on Streams or ChangeNotifier, instead, it uses low latency GetValue and GetStream to improve performance. It also facilitates communication between the client and server through HTTP requests.

4.3.4 Google Cloud Platform

Google Cloud Platform (GCP) is a cloud computing platform offered by Google that provides a wide range of cloud-based services and tools to help individuals and organizations build, manage, and deploy applications and services. Such as virtual machines. You can also provide access to Google products using APIs, such as accessing Drive to get patient data (drug names).

4.4 Mobile development

In our App we provide you can set which language will use in the App, and you can change it again many times after creating an account.



Figure 4.1: Splash Screen



Figure 4.2: Onboarding Screen 1

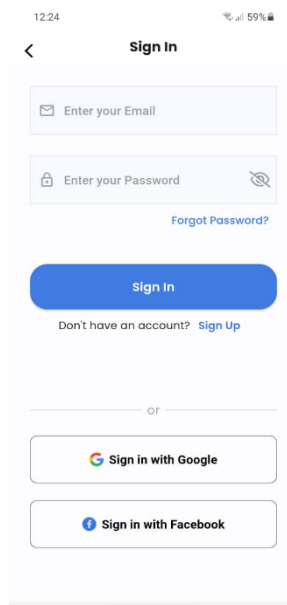


Figure 4.3: Onboarding Screen 2

- First, you have to sign up.

Figure 4.4: sign up

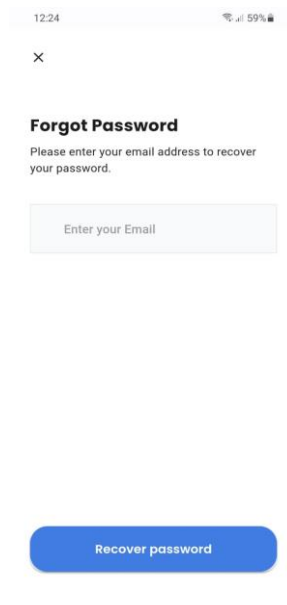
- If you already have an account, you can login, and you can login with google or Facebook



The screenshot shows a mobile app interface for signing in. At the top, the status bar displays the time 12:24 and battery level at 59%. The app header includes a back arrow and the title "Sign In". Below the header are two input fields: "Enter your Email" with an envelope icon and "Enter your Password" with a lock icon and a toggle for visibility. A "Forgot Password?" link is positioned below the password field. A prominent blue "Sign In" button is centered below the inputs. Underneath the button, a link reads "Don't have an account? Sign Up". A horizontal separator with the text "OR" is placed below the main sign-in section. At the bottom, there are two buttons for social login: "Sign in with Google" and "Sign in with Facebook".

Figure 4.5: Login

- Password: Our App helps you if forgetting password by entering the address associated with your account then you can reset your password.



The screenshot displays the "Forgot Password" screen in the mobile app. The status bar at the top shows 12:24 and 59% battery. The app header features a close icon (X) and the title "Forgot Password". Below the title, a message states: "Please enter your email address to recover your password." This is followed by an "Enter your Email" input field. At the bottom of the screen is a blue button labeled "Recover password".

Figure 4.6: Reset Password

The “Profile Edit” screen is a critical component of our app, designed to be accessed immediately after logging in. It allows users to personalize their experience by selecting their role as either a “Normal patient” or “Care giver”. Users can also manage their profile picture and contact information, and set specific times for their main meals. These settings are crucial for tailoring the app’s functionality to the user’s needs. All changes can be saved by clicking the “Save Changes” button at the bottom of the screen. This screen underscores our app’s commitment to providing a user-friendly and personalized experience for Alzheimer’s patients and their caregivers.



Figure 4.7: Account Screen

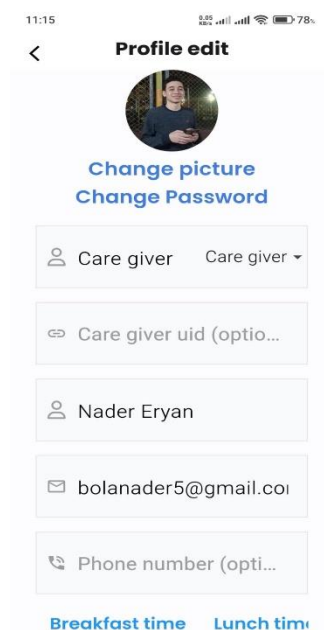


Figure 4.8: Edit Profile

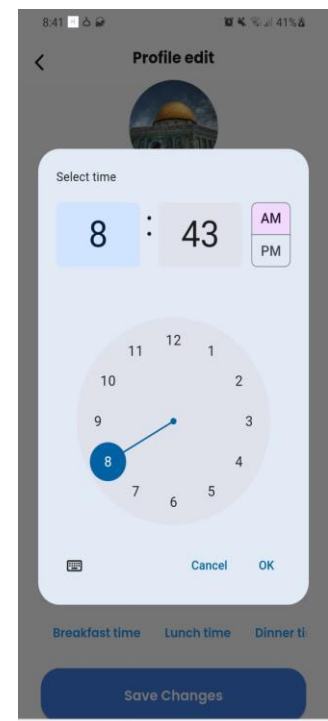


Figure 4.8: Select Meal Time

1. **Medication Input:** The screen allows users to add a new medicine to their schedule. Users can enter the name of the medicine in the “Medicine” field. In your example, “Leptirodin” has been entered.
2. **Medication Timing:** Users can specify when the medicine should be taken relative to their meals. There are four options available: “Before breakfast,” “After breakfast,” “Before lunch,” and “After lunch.” This feature is crucial for ensuring that medication is taken at the most effective times.
3. **Save Medication Record:** After entering all the necessary information, users can add the medication record to their profile by clicking the “Add Medical Record” button at the bottom of the screen.

5:31 20%

< Add Medication

See All

Lepirudin

Extra note (optional)

When the drug is taken

☒ Before breakfast ☐ After breakfast

☐ Before lunch ☐ After lunch

☐ Before dinner ☐ After dinner

Add Medical

Figure 4.9: Add New Medicine

When the user clicks on “See All” in the app, they are taken to the “Medication Types” screen. This screen is designed to help users manage their medication more effectively by categorizing them into six different types: Tablet, Drop, Cream, Solution, Injection, and Inhaler. Each type is represented by an icon for easy identification. This screen helps users manage their medication schedules effectively, ensuring correct and consistent medication intake.

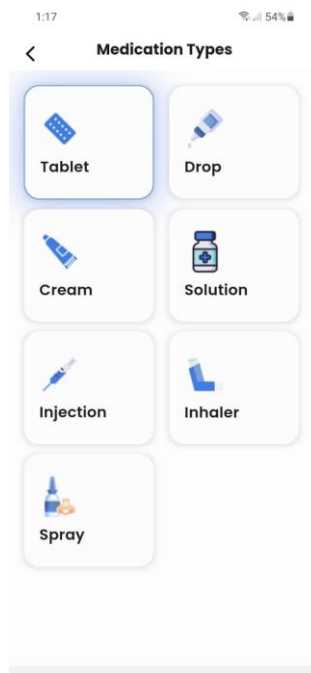


Figure 4.10: Select Types

1. **Active Medications:** The “Active meds” section lists the medications that are currently being taken by the user. Each medication entry shows the name of the medication and the time it should be taken. In example, “Treprostinil” is to be taken before and after breakfast, and “Lepirudin” is to be taken before breakfast
2. **Inactive Medications:** The “Inactive meds” section would list medications that the user is not currently taking. In your example, there are no inactive meds.
3. **Editing Options:** Users can interact with each medication entry. They can delete a medication or move it to the “Inactive meds” section by pressing the “Delete” or “Correct” button respectively.

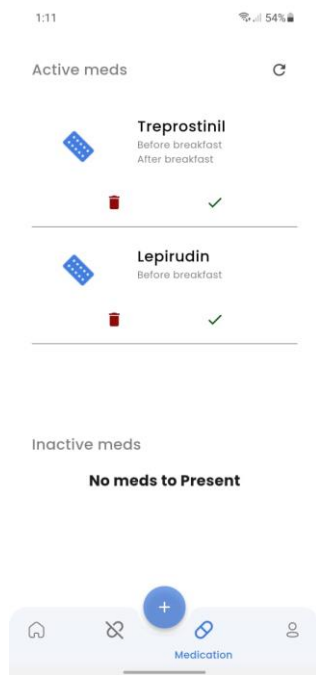


Figure 4.11: Active Medications

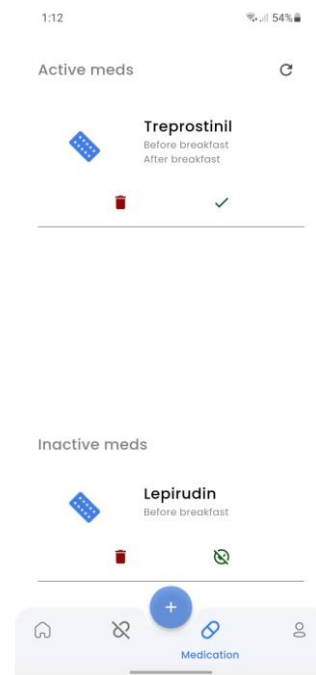


Figure 4.12: Inactive Medications



Figure 4.13: Drug Drug Interaction



Figure 4.14: Drug Drug No Interaction

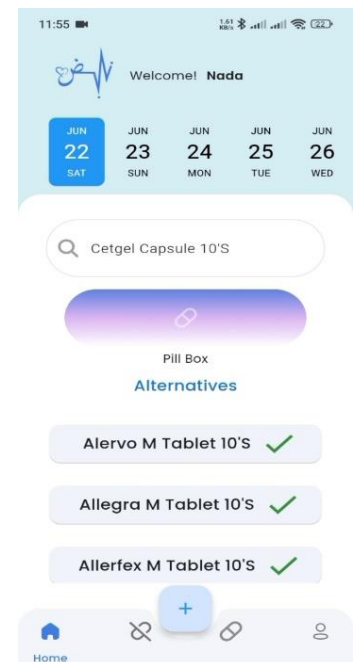


Figure 4.15: Medications Alternatives

Chapter 5

Drug-Drug Interaction and Drugs Recommendation

5.1 Experimental Results and Comparative Analysis to Drug-Drug Interaction (DDI) Model

5.1.1 Experiment Specifications and Used Materials

Data Source

- DrugBank Database: The data is sourced from the DrugBank database, which contains comprehensive information on drugs. This dataset includes details on approximately 12,000 drugs, their DrugBank IDs, names, and drug interactions.

Libraries and Tools Used

1. Pandas: A powerful data manipulation and analysis library in Python. It is used here to read the data from an Excel file and perform various data operations.
2. Regex: The regex library is used to apply regular expressions for extracting DrugBank IDs from the drug interactions data.
3. NetworkX: A Python library for creating, manipulating, and studying complex networks of nodes and edges. It is used here to construct and analyze the drug interaction network.
4. Matplotlib: A plotting library for the Python programming language and its numerical mathematics extension NumPy. It is used to visualize the drug interaction network.

5.1.2 Evaluation Metrics

1. Drug Name Matching: The effectiveness of matching user-entered drug names with those in the dataset. This involves case-insensitive string comparison.
2. Drug Interaction Identification: The identification of interactions between user-entered drugs by checking if their DrugBank IDs appear in the drug-interactions column for other user-entered drugs.
3. Visualization: The clarity and comprehensiveness of the resulting drug interaction network graph.

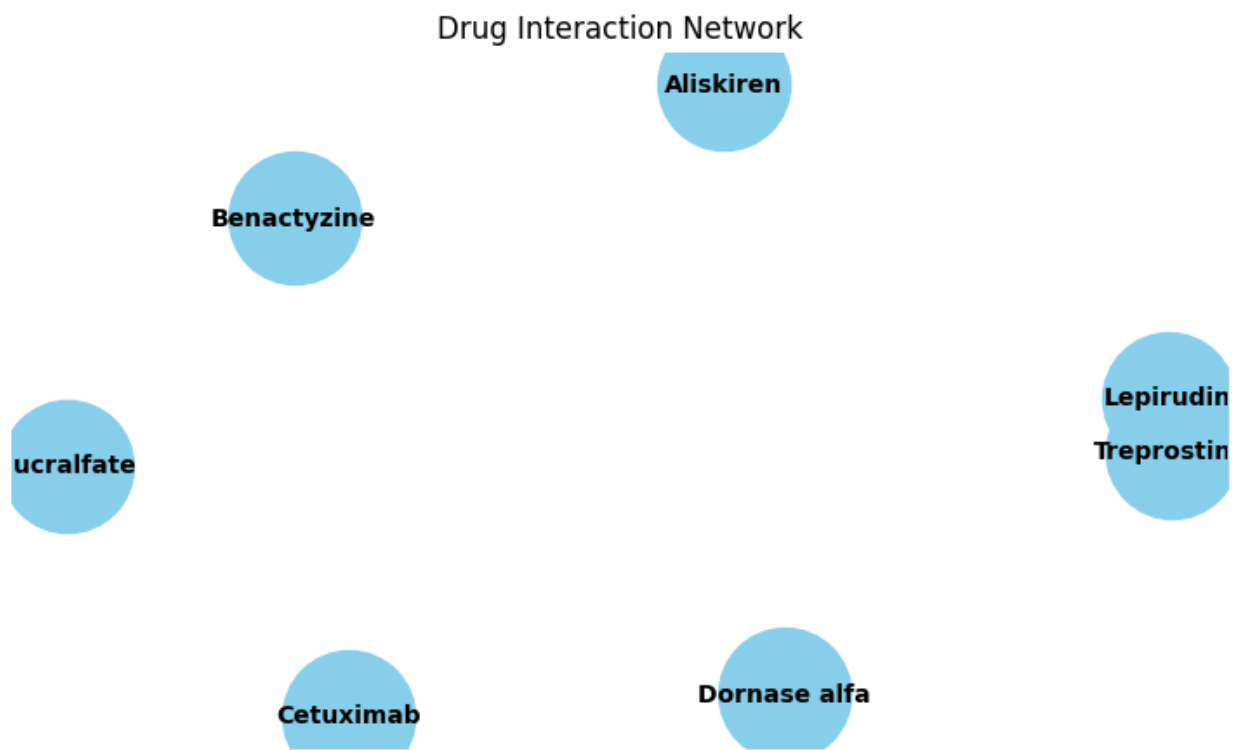


Figure 4.1: Drug Interaction Network

5.1.3 Results of The System

5.1.3.1 Dataset Description

- DrugBank ID: A unique identifier assigned to each drug in the DrugBank database.
- Drug Name: The common name of the drug.
- Drug Interactions: A column containing DrugBank IDs of drugs that interact with the given drug, formatted as a comma-separated string.

The dataset used in this experiment includes three columns: drugbank-id, name, and drug-interactions.

	drugbank-id	name	drug-interactions
0	DB00001	Lercanidipine	DB01381,DB00374
1	DB00002	Cetuximab	
2	DB00003	Dornase alfa	
3	DB00004	Denileukin diftitox	DB06372,DB00072
4	DB00005	Etanercept	DB01281,DB00026,DB08879,DB08879,DB06168,DB0890...

Figure 4.2: Dataset DDI

5.2 Experimental Results and Comparative Analysis to Medicine Recommendation System Model

5.2.1 Experiment Specifications and Used Materials

Data Source

The dataset is sourced from Kaggle, a platform for data science and machine learning competitions. The specific dataset used in this experiment contains information about medicines, including their names, descriptions, and reasons for use.

Libraries Used

Similar to the previous explanation, the code likely uses libraries such as numpy, pandas, nltk, and sklearn for data manipulation, natural language processing, and machine learning tasks.

5.2.2 Evaluation Metrics

- **Data Cleaning:** The code likely performs data cleaning operations, such as removing null values, duplicates, and unnecessary characters.

index		Drug_Name	Reason	Description
0	1	A CN Gel(Topical) 20gmA CN Soap 75gm	Acne	Mild to moderate acne (spots)
1	2	A Ret 0.05% Gel 20gmA Ret 0.1% Gel 20gmA Ret 0...	Acne	A RET 0.025% is a prescription medicine that i...
2	3	ACGEL CL NANO Gel 15gm	Acne	It is used to treat acne vulgaris in people 12...
3	4	ACGEL NANO Gel 15gm	Acne	It is used to treat acne vulgaris in people 12...
4	5	Acleen 1% Lotion 25ml	Acne	treat the most severe form of acne (nodular ac...

Figure 4.3: Dataset Medicine Recommendation

- **Text Preprocessing:** Text preprocessing tasks, such as converting text to lowercase, tokenization, and stemming, are likely applied to prepare the text data for analysis.

index		Drug_Name	tags
0	1	A CN Gel(Topical) 20gmA CN Soap 75gm	mild to moderate acne (spots) acne
1	2	A Ret 0.05% Gel 20gmA Ret 0.1% Gel 20gmA Ret 0...	a ret 0.025% is a prescription medicine that i...
2	3	ACGEL CL NANO Gel 15gm	it is used to treat acne vulgaris in people 12...
3	4	ACGEL NANO Gel 15gm	it is used to treat acne vulgaris in people 12...
4	5	Acleen 1% Lotion 25ml	treat the most severe form of acne (nodular ac...

Figure 4.4: Dataset after preprocessing

- **Similarity Calculation:** Cosine similarity or another similarity metric is likely used to calculate the similarity between medicines based on their descriptions and reasons for use.

```
array([[1.          , 0.25197632, 0.43643578, ..., 0.          , 0.          ,
        0.          ],
       [0.25197632, 1.          , 0.25660012, ..., 0.19245009, 0.1490712 ,
        0.0860663 ],
       [0.43643578, 0.25660012, 1.          , ..., 0.11111111, 0.0860663 ,
        0.0993808 ],
       ...,
       [0.          , 0.19245009, 0.11111111, ..., 1.          , 0.77459667,
        0.2981424 ],
       [0.          , 0.1490712 , 0.0860663 , ..., 0.77459667, 1.          ,
        0.34641016],
       [0.          , 0.0860663 , 0.0993808 , ..., 0.2981424 , 0.34641016,
        1.          ]])
```

Figure 4.5: Vectors to dataset descriptions (tags)

- **Recommendation Accuracy:** The system's accuracy in recommending similar medicines is likely evaluated based on the cosine similarity scores or other similarity metrics.

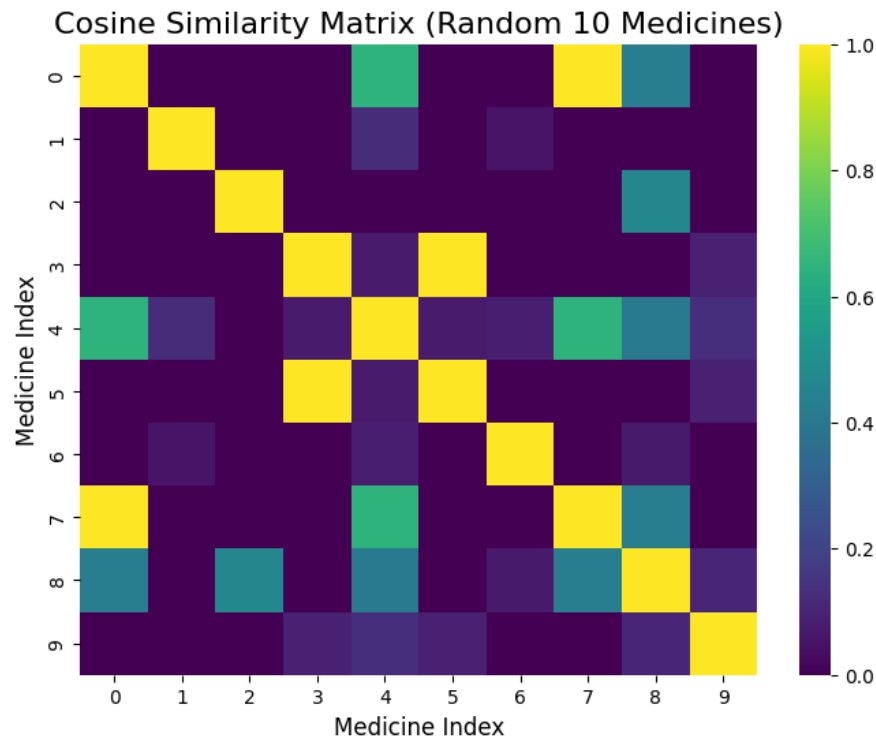


Figure 4.6: Similarity matrix to medicine

5.2.3 Results of The System

Dataset Description

- **Columns:** The dataset likely contains columns such as 'index', 'Drug_Name', 'Description', and 'Reason'.
- **Description:** Describes the medicine in terms of its composition, dosage, and other relevant information.
- **Reason:** Specifies the medical condition or reason for which the medicine is prescribed.

Results and Discussion

- **Data Cleaning:** Null values and duplicates are likely removed to ensure data quality.
- **Text Preprocessing:** The text data is likely preprocessed to convert it into a format suitable for analysis, including lowercasing, tokenization, and stemming.
- **Similarity Calculation:** Cosine similarity or another similarity metric is likely calculated to determine the similarity between medicines based on their descriptions and reasons for use.

- **Recommendation System:** The system likely recommends similar medicines based on the input medicine, demonstrating the effectiveness of the cosine similarity approach or other similarity metrics.

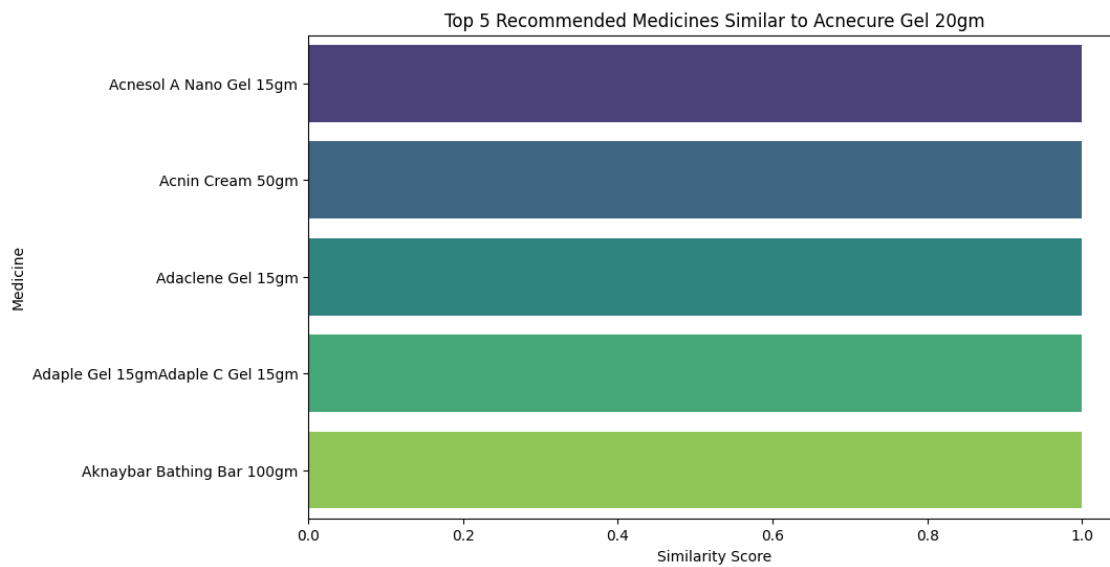


Figure 4.7: Similarity Score To Recommended Medicines

Chapter 6

Embedded System

6.1 Overview

An embedded system is a combination of computer hardware and software designed for a specific function and it is a specialized computer system designed to perform specific tasks within larger devices or machines, optimized for reliability, real-time operation, and low power consumption.

6.2 Hardware components

- ESP32
- Servo Motor
- Buzzer
- LEDs
- Box

6.2.1 ESP32

The ESP32 microcontroller from Espressif Systems is known for its rich set of features, which include:

1. **Dual-core Processor:** It features a dual-core 32-bit Tensilica LX6 processor, which allows for efficient multitasking and handling of complex tasks.
2. **Wi-Fi Connectivity:** Integrated 2.4 GHz Wi-Fi (802.11 b/g/n) supporting WPA/WPA2 security protocols, making it suitable for wireless connectivity in IOT applications.
3. **Bluetooth Connectivity:** Built-in Bluetooth Low Energy (BLE) support, enabling communication with other Bluetooth devices and peripherals.
4. **Low Power Consumption:** Optimized power management capabilities, making it suitable for battery-operated devices and applications requiring energy efficiency.
5. **Rich Peripheral Interface:** Includes a wide range of peripheral interfaces such as SPI, I2C, UART, ADC, DAC, PWM, and GPIOs, facilitating connectivity with sensors, displays, actuators, and other components.
6. **Security Features:** Supports secure boot, flash encryption, and cryptographic hardware acceleration, enhancing data security in IOT applications.

We Use ESP32 To:

1- Control and Automation:

- The ESP32 manages the physical pill dispenser, controlling the release of pills based on the schedule configured through the mobile application.
- It ensures accurate and timely dispensing according to the user's medication schedule.

2- Communication Interface using Wi-Fi Connectivity:

- Acting as a bridge between the physical dispenser (box) and the mobile application, the ESP32 facilitates communication.
- It receives commands from the application and sends status updates back to the application.

3- Data Synchronization:

This synchronization ensures that the user has up-to-date information and control over their medication management through the mobile app.

4- User Interface Support:

The ESP32 interface with user interface elements on the dispenser (such as LEDs, Buzzer) and synchronize this information with the mobile application.

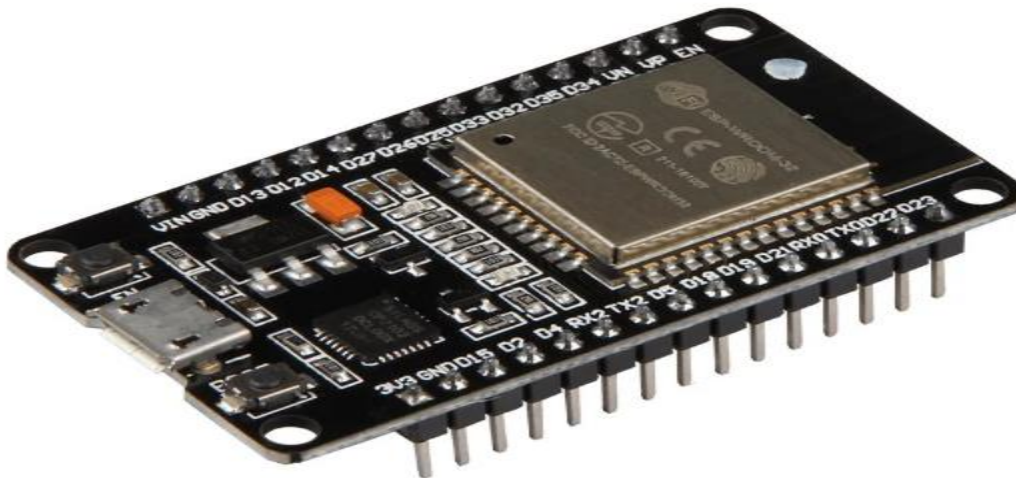


Figure 6.1: ESP32

6.2.2 Servo Motor

A servo motor is a precise rotary actuator that uses feedback to control angular position, velocity, and acceleration. It consists of a DC motor, gears, and a feedback mechanism for accurate positioning in various applications such as robotics, automation, and consumer electronics.

We use it to control the opening of doors:

- 1- The medicine slot door at the time of taking the medicine, so that it falls into the drawer designated for the patient so that he can take the medicine from it at the specified time.
- 2- An upper door opens for him Caregiver to fill the slots when the box becomes empty.



Figure 6.2: Servo Motor

6.2.3 Buzzer

A buzzer is an electronic device that produces a buzzing or beeping sound, commonly used for alerts, notifications, and signaling purposes in electronic circuits and devices.

Therefore, we use it to issue an alert sound at the time of medication so that the patient pays attention to taking the medication.



Figure 6.3: Buzzer

6.2.4 LEDs

LEDs, or Light-Emitting Diodes, are semiconductor devices that emit light when an electric current passes through them. They are used for indicators, displays, and lighting due to their efficiency, longevity, and compact size.

We use LEDs that light up at the time of medication and are located in each slot so that the patient knows which medication he is taking.



Figure 6.4: LEDs

6.2.5 Box

Box is a physical pill dispenser unit to dispense pills accurately according to the schedule set on the app and alerts back to patient to take medicine at its time.



Figure 6.5: Box interface

It Includes

- 7*6 slots, which includes 7 days, each day 6 times to take the medicine before and after each meal.
- Upper doors which open using a servo motor when the caregiver fill the slots with medicine when its became empty.



Figure 6.6: Box Slots & Upper Doors

- Drawer: When the pills fall out of their slots using servo motor at the time of taking the medicine It falls into that drawer, and then the patient opens that drawer to take his medicine.



Figure 6.7: Box Drawer

Chapter 7

Conclusions & Future Work

7.1 Conclusions

In conclusion, the development of the “Pulse” medication management application has been driven by the necessity to assist patients, particularly those with chronic conditions or cognitive impairments, in managing their medications effectively and safely. The application combines a user-friendly interface, a robust architecture using Flutter and Dart, and an AI model to ensure drug safety and interaction checks.

The physical pill dispenser adds an additional layer of convenience and adherence by dispensing the correct medication at the right time. The integration of features such as reminders, allergy tracking, and caregiver supervision ensures comprehensive support for patients, reducing the burden on caregivers and enhancing the overall healthcare experience

7.2 Future work

For future work we aim to:

1. Integration with doctors and Pharmacies
2. Patient and Caregiver Education to inform them about the importance of medication adherence, potential side effects, and lifestyle choices that can impact health.
3. Increase amount of dataset used to cover all possible drugs.
4. We will keep each type of drugs in the same slot, and when the caregiver enters the drugs and its dates, we will collect the quantity of the drugs at the appropriate time and give it to the patient.