



## **FLUTTER MEDICINE TRACKER AND NOTIFICATION APP**

A graduation project report submission  
In partial fulfillment of the requirements for the award of the degree  
Bachelor of Science

Submitted by:

Mark Emad	89603
Yossef Mohamed	89551
Seif Hany	89568
Youssef Khaled	94246

Under the supervision of professor:

Dr. Alaa Zaghloul

TA. Eng. Rahma Waleed & Eng. Omar Ayman

Department of Computer Science - CS  
Misr University for Science and Technology - MUST  
College of Computers and Artificial Intelligence Technology - CAIT

## **Acknowledgments**

First, we are very grateful to Almighty Allah who gave us opportunity, strength, determination and wisdom to achieve our goal. Without His support, this could not have been possible.

Though only our names appear on the cover of this dissertation, a great many people have contributed to its production. We owe our gratitude to all those people who have made this dissertation possible and because of whom our graduate experience has been one that we will cherish forever.

Foremost, we would like to express our sincere gratitude to our advisor Dr. Alaa Zaghloul for the continuous support of our project study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in all the time of research and writing of this research. We could not have imagined having a better advisor and mentor for our project study. We also wish our sincere thanks and appreciation to Dean. Dr. Rania Elgohary for her support of us all the time.

Besides our advisors, we would like to thank the rest of our research committee: Eng. Rahma Waleed and Eng. Omar Ayman for their expertise, understanding, and patience, that added considerably to our graduate experience. We appreciate their vast knowledge and skill in many areas (e.g., vision, aging, ethics, interaction with participants), and their assistance in writing reports (i.e., grant proposals, scholarship applications and this thesis), which have on occasion made us "GREEN" with envy.

Finally, most importantly, none of this would have been possible without the love and patience of our families. Our immediate families to whom this dissertation is dedicated to, has been a constant source of love, concern, support and strength all these years. We would like to express our heart-felt gratitude to our families.



## **Declaration**

I hereby certify that this work, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in (insert title of degree for which registered) is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others and to the extent that such work has been cited and acknowledged within the references section of this report.

Signed: \_\_\_\_\_

Registration No.: \_\_\_\_\_

Date:



### **Abstract:**

The primary aim of our project is to develop a robust and all-encompassing medicine reminder application that prioritizes the individual needs and preferences of users. Central to our vision is the creation of a user-friendly interface that effortlessly allows patients to input and diligently track crucial medical information, encompassing medication schedules, dosages, frequencies, and refill requirements. By leveraging the power of our application, the burdensome task of remembering specific doses for each medication is alleviated, offering patients a simplified and stress-free experience.

Our application is designed with utmost accessibility in mind, ensuring that individuals from all walks of life, including patients, nurses, and family members, can actively engage in their health management journey. By providing a comprehensive platform that grants users access to accurate and up-to-date information anytime and anywhere, we empower them to take charge of their health and make informed decisions regarding their well-being.

Through our project, we envision a future where medication management transcends the realm of complexity and becomes a seamless and intuitive process for all users. By merging cutting-edge technology with user-centric design principles, our application sets forth a new standard in empowering individuals to proactively engage with their health, promoting optimal medication adherence and ultimately fostering improved health outcomes.

## Table of Contents

<b>LIST OF FIGURES .....</b>	<b>II</b>
<b>LIST OF TABLES .....</b>	<b>III</b>
<b>1 INTRODUCTION .....</b>	<b>1</b>
1.1 Overview .....	1
1.2 Problem Definition .....	3
1.3 Project Objectives .....	4
<b>2 LITERATURE SURVEY AND RELATED WORK .....</b>	<b>5</b>
2.1 Related work .....	5
2.2 Comparative study .....	15
2.3 Expected outcomes .....	16
<b>3 SYSTEM ANALYSIS AND DESIGN .....</b>	<b>17</b>
3.1 Context Diagram .....	17
3.2 Data flow Diagram .....	18
3.3 Sequence Diagram .....	20
3.4 State chart .....	23
3.5 Use case Diagram .....	25
3.6 Class Diagram .....	26
3.7 Entity relationship Diagram .....	27
<b>4 PROPOSED SYSTEM .....</b>	<b>28</b>
4.1 System architecture .....	28
4.2 Algorithms and methodologies .....	29
4.3 Proposed software and hardware .....	34
4.4 Functional requirements .....	35
4.5 Nonfunctional requirements .....	35
<b>5 RESULTS AND DISCUSSION .....</b>	<b>36</b>
<b>REFERENCES .....</b>	<b>43</b>
<b>APPENDICES .....</b>	<b>44</b>
Appendix A: Code used for development .....	44
Appendix B: Algorithm used .....	59

## List of Figures

Figure 2-1: Medslog app.....	5
Figure 2-2: Medhelper app.....	7
Figure 2-3: Dosecast app.....	7
Figure 2-4: Mango health app.....	8
Figure 2-5: Everydose app.....	10
Figure 2-6: Mytherapy pill app.....	10
Figure 2-7: Medisafe app.....	13
Figure 4-1: System architecture.....	28
Figure 4-2: Rapid Application Development.....	30
Figure 5-1: Launch screen.....	37
Figure 5-2: Home page.....	38
Figure 5-3: Add medicine page.....	39
Figure 5-4: Text recognition page.....	40
Figure 5-6: Export medicine list page.....	41



## LIST OF TABLES

Table 2-1: Comparative study .....	15
------------------------------------	----



## *C H A P T E R O N E*

# **1 INTRODUCTION**

### **1.1 Overview**

In today's fast-paced and increasingly complex world, where managing multiple responsibilities is a common challenge, keeping track of medication schedules has become even more daunting for individuals. This is especially true for those with chronic illnesses or those who rely on multiple medications to maintain their health and well-being. The potential consequences of missing a crucial dose or accidentally exceeding the prescribed dosage can be dire, ranging from adverse drug reactions to hospitalization, underscoring the critical need for a reliable solution.

Recognizing this pressing issue, our medicine tracker mobile application emerges is for individuals seeking effective medication management. By seamlessly integrating advanced technological solutions with intuitive user experience, our application is purposefully designed to empower users in efficiently and effortlessly managing their medication schedules. With a holistic approach to enhancing overall health and well-being, our app serves as a dependable companion, lending support and alleviating the burden of medication management.

At the core of our medicine tracker mobile application lies a user-friendly interface, designed to cater to the diverse needs and preferences of our valued users. Through its intuitive design, navigating the app becomes an intuitive and stress-free experience, ensuring that users can effortlessly use its features. One of the key functionalities offered by our app is its intelligent medication reminder system, which diligently alerts users when it is time to take their prescribed medications. By delivering timely and personalized reminders, our application mitigates the risk of missed doses, thus promoting adherence to medication regimens and minimizing potential health complications.

Furthermore, our application incorporates a sophisticated text recognition feature, revolutionizing the way users interact with their medication information. Through the app's





seamless integration with optical character recognition (OCR) technology, users can effortlessly capture and digitize important details from prescription labels, medication packaging, or other relevant documents. This enables the app to automatically populate the medication profiles with accurate information, ensuring that users have a comprehensive and up-to-date overview of their prescribed treatments. By streamlining the process of entering medication details, our application simplifies the initial setup and ongoing management of medication profiles, ultimately enhancing the overall user experience.

In addition to its medication management capabilities, our application goes above and beyond by incorporating a thoughtful refill reminder system. Understanding the importance of timely medication refills, our app diligently monitors the user's medication supply and proactively notifies them when it is time to replenish their stock. Our application takes the hassle out of managing medication refills, ensuring that users never run out of essential treatments.

Through its comprehensive and user-centric approach, our medicine tracker mobile application emerges as an indispensable tool in empowering individuals to take control of their medication schedules effectively. By seamlessly integrating features such as medication reminders, advanced text recognition, and timely refill reminders, our application not only simplifies the process of managing medications but also instills a sense of confidence and peace of mind in users. By optimizing medication adherence and enhancing overall health outcomes, our app serves as a steadfast companion on the journey toward improved well-being and a better quality of life.



## **1.2 Problem Definition:**

Medication non-adherence poses a significant challenge, contributing to adverse health consequences and escalating healthcare expenditures. The reasons behind non-adherence are multifaceted, encompassing forgetfulness, misconceptions regarding medication instructions, and the occurrence of unpleasant side effects. The root of the issue that our medicine tracker application addresses lies in the complexities individuals encounter when juggling multiple medications. This includes the struggle to remember when to take each medication, the correct dosages, and keeping track of whether a dose has already been taken.

Our medicine tracker mobile application helps by offering a comprehensive solution to empower users in effectively managing their medication schedules and bolstering adherence to prescribed regimens. By seamlessly integrating advanced technology with user-centric design principles, our application strives to eliminate the possibility of human error while tracking medications, ultimately enhancing the safety and efficacy of medication management.

However, with a myriad of medicine tracker apps saturating the market, finding one that precisely caters to the unique needs and preferences of each user can be a hard task. Moreover, some individuals may find the process of inputting medication data and configuring reminders to be time-consuming or bewildering, further hindering their adherence efforts. Therefore, the underlying problem that our medicine tracker mobile application aims to tackle revolves around providing a platform that is both user-friendly and highly customizable, ensuring that users can effortlessly manage their medication schedules in a manner that seamlessly integrates into their daily lives. By streamlining the user experience and offering extensive customization options, our application aims to alleviate the burden of medication management while enhancing overall health outcomes.

By addressing the challenges associated with medication non-adherence, our medicine tracker app strives to contribute to improved health outcomes, reduced healthcare costs, and ultimately enhance the quality of life for individuals managing complex medication regimens.



### **1.3 Project objectives:**

Develop an application that has the following features:

1. Remind patients to take their medications on time.
2. Remind patients of the dose and the frequency for each medicine.
3. Remind patients to refill their medications before they run out.
4. Text Recognition feature.
5. Export Medication Log.
6. Remind patients of their doctor appointments and visits.
7. Send SOS emergency alerts to family members.
8. Look up detailed information about any registered medication.
9. Store medicine and its information in a database, if a user uses the same medicine more than one time.

## CHAPTER TWO

# 2 LITERATURE SURVEY AND RELATED WORK

## 2.1 Related Work

All the applications that are going to be mentioned work the same way but differ in small things like the user interface and usability for most of the apps. The common features of all applications are: you type in the name of your medications, the dose for each medicine, how often you take it, and when you'd like to be reminded to take it.

- [MedsLog](#)

The Medslog app is an iPhone application specifically designed to assist patients in managing their medication schedules effectively. It serves as a reliable reminder system, ensuring that individuals never miss a dose. In fact, a user of the Medslog app shared their positive experience in an article, highlighting the challenges they faced in remembering the frequency of their medication intake. They often found themselves inadvertently taking more pills than necessary. However, with the Medslog app, they were able to delegate the responsibility of tracking their medication schedule to the application. The user found the app to be immensely helpful in managing their medications, providing them with a sense of relief and peace of mind.

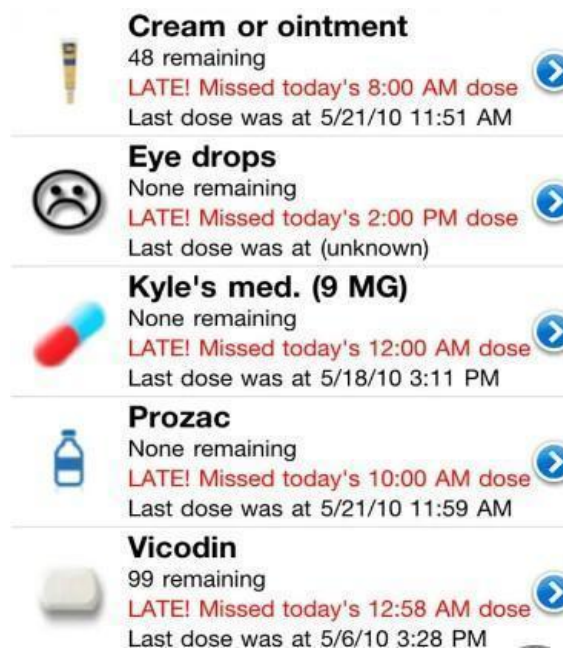


Figure 2- 1

It is worth noting that the Medslog app offers a lite version priced at \$4, which allows users to access its core features. However, this cost can be considered a minor drawback for some users. Nonetheless, the app's ability to effectively address the issue of medication adherence outweighs this limitation. By streamlining the medication management process, the Medslog



app proves to be an invaluable tool for individuals seeking to improve their adherence to prescribed medication regimens.

While Medslog initially appeared to be user-friendly, the user's experience with the app gradually changed over time. They encountered challenges primarily related to the app's user interface (UI), which they found to be difficult to navigate. Additionally, they faced numerous issues while utilizing the app. Moreover, Medslog is exclusively available for iPhone users, which restricts access for Android users. This limitation arises from the inherent complexity of iOS compared to Android, making it more challenging to establish seamless app integrations. In the case of Medslog, it required linking with the iPhone calendar to set medication alarms.

Unfortunately, Medslog may not be the most suitable choice for elderly individuals due to its complex and hard-to-use interface. This aspect should be considered when contemplating its usage for grandparents or individuals with limited technological experience. Furthermore, for individuals with busy schedules, such as those working full-time jobs, the app may present a significant learning curve, leaving little time to dedicate to understanding its functionality or teaching it to older family members. Consequently, opting for Medslog in such circumstances may prove to be an impractical decision.

- [Medhelper](#)

Medhelper application is an iOS and android application, helps people manage their care plan, the app uses a flexible and friendly interface.

The app works in an easy way, the user adds his medicine, selects the time to take and the frequency that's all.

The application also stores medicines names in a database, so the user searches for his medicine, see some information about it as the side effects of the medicine.

The user receives notification at the medicine time, reminding them “that’s the medicine time “.

Besides that, the user can add notes about their medicine or the overall care plan.

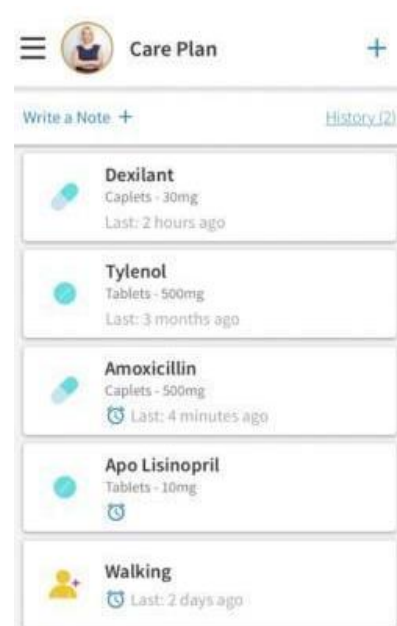


Figure 2- 2

- [Dosecast:](#)

Dosecast, similar to Medslog, is a highly regarded mobile application widely recognized for its functionality. While the app itself is available for free, there are additional premium features that can be unlocked through in-app purchases. It is worth noting that



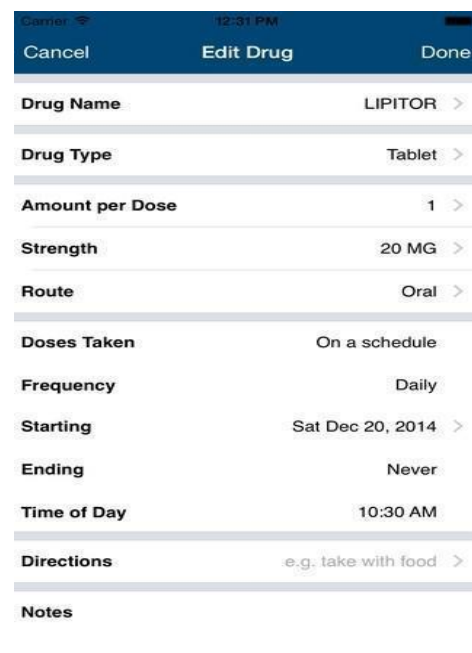
Figure 2- 3

Dosecast shares a common drawback with Medslog in terms of its cost, requiring users to invest \$4 for full access. Although comparatively cheaper than Medslog, this payment requirement may limit its accessibility to a certain extent.

However, Dosecast distinguishes itself by offering a more user-friendly experience. Users have found it to be intuitive and easy to navigate, enhancing their overall satisfaction with the app. Unfortunately, one notable downside of Dosecast is the absence of an auto-fill feature for medication names. Consequently, users are required to manually input the complete name of each medicine they need to track, which can be a tedious process.

Despite this drawback, Dosecast boasts an exceptional feature that sets it apart—the app's notifications function seamlessly even when the iPhone is locked or when another app is running. This means that users can rely on timely reminders without the need for an active cellular data connection. Moreover, the app's notification system persists until the user acknowledges it, even if the sound on the device is turned off, as the phone vibrates until the user manually dismisses the notification.

In conclusion, Dosecast is a highly rated medication reminder app that offers valuable features. While the app itself is free to download, there are premium features available through in-app purchases. Although it shares a common cost-related limitation with Medslog, Dosecast distinguishes itself by providing a user-friendly interface. However, users should be aware that it lacks an auto-fill feature for medication names, requiring manual entry. Nevertheless, Dosecast excels in delivering reliable notifications, even when the iPhone is locked or when other apps are in use, making it a convenient tool for medication management.



- **Mango Health:**

This particular app is widely recognized as one of the prominent medicine reminder applications available. It offers a range of features and stands out as a free app that enables users to conveniently set medication reminders. However, its utility extends beyond medication reminders, as it also provides reminders for various other healthy habits.



Figure 2- 4



In addition to medication reminders, this app offers the flexibility to customize reminders for healthy activities tailored to individual needs. For instance, it allows users to set reminders for monitoring blood sugar levels or tracking water intake. Such personalized reminders can prove invaluable in maintaining a consistent and healthy routine.

Apart from reminders, the app includes a health diary feature that allows users to log upcoming appointments and laboratory tests. This feature enhances organization and facilitates better healthcare management. Furthermore, the app provides users with valuable information on medication interaction warnings and potential side effects, empowering them to make informed decisions about their health.

Another noteworthy aspect of this app is its ability to notify users when medication refills are needed. This feature ensures that users never run out of essential medications and helps streamline the medication management process.

One particularly intriguing feature of this app is its points and rewards system. Users can earn points for consistently adhering to their medication schedules, and these points can be redeemed for various gift cards. This gamification element adds an element of motivation and reward to medication adherence, potentially increasing user engagement and commitment.

In summary, this app is renowned as a comprehensive medicine reminder application. It not only offers medication reminders but also includes reminders for various other healthy habits. With features such as customized healthy activity reminders, a health diary, medication interaction warnings, refill notifications, and a points and rewards system, the app provides users with a holistic approach to managing their health. By combining functionality with gamification, it encourages users to stay on track with their medication routines while also promoting overall well-being.



- **EveryDose:**

EveryDose, similar to the aforementioned apps, is a free application that offers customizable medication reminder features, alerts and comprehensive reports to facilitate effective medication management. It empowers users to stay on top of their medication schedules and ensures timely adherence.



Figure 2- 5

What sets EveryDose apart from the other apps is the presence of a unique feature called Maxwell, a virtual assistant specifically designed to enhance the user experience. Maxwell serves as a valuable resource, enabling users to seek answers to medication-related inquiries. With this feature, users can gain a better understanding of the medications they are taking, including their purposes, potential side effects, and any other concerns they may have. Maxwell's assistance adds an additional layer of support and education to the app, fostering informed decision-making and promoting medication safety.

By combining the core functionalities of customizable medication reminders, alerts, and comprehensive reports with the innovative inclusion of Maxwell, EveryDose ensures that users have the tools and knowledge necessary to manage their medications effectively. It goes beyond being a simple reminder app by providing an interactive and informative experience that empowers users to take control of their health and medication regimens.

- **MyTherapy Pill:**

The MyTherapy Pill Reminder app, similar to the other applications mentioned, offers users the ability to create personalized medication reminders to ensure timely intake. Additionally, it provides a logbook feature that allows users to track their medication intake and share this information with others involved in their healthcare journey.



Figure 2- 6

One distinguishing feature of the MyTherapy app is its capability to track the specific injection sites for medications, such as the upper arm or stomach. This feature caters to



individuals who require injections as part of their treatment regimen, providing them with a convenient way to monitor and record the injection locations.

Moreover, the MyTherapy app extends beyond medication tracking by offering the option to monitor various other health-related information. Users can track their mood, weight, and blood pressure, among other measurements, which contributes to a comprehensive overview of their overall well-being. This holistic approach to health tracking allows users to observe patterns, identify correlations, and gain insights into the impact of their treatment and lifestyle choices.

MyTherapy stands out as an ad-free, award-winning pill reminder and medication tracker. However, it offers more than just medication reminders. By integrating a mood tracker, health journal, and other measurement tracking features, such as weight and blood pressure, the app provides users and their healthcare providers with a holistic perspective on treatment effectiveness. The app's versatile features cater to a wide range of conditions, including diabetes, rheumatoid arthritis, anxiety, depression, hypertension, and multiple sclerosis, making it a valuable tool for individuals managing various health concerns.

In summary, the MyTherapy Pill Reminder app aligns with other apps in terms of customizable medication reminders and logbook functionality. However, it distinguishes itself by allowing users to track injection sites and offering comprehensive health tracking options, including mood, weight, and blood pressure. The app's features aim to provide a holistic perspective on treatment success, allowing users and their doctors to evaluate progress and make informed decisions.

- **[Cute Pill: Medication Reminder](#)**

Key Features:

1. **Simplicity and Ease:** The app offers a user-friendly interface that is simple and easy to navigate, ensuring a hassle-free experience.
2. **No Membership Registration:** Users can access all the features without the need for cumbersome registration or membership processes.



3. Medication Recording: Users can effortlessly record whether they have taken or used their medications, providing an accurate log of their medication intake.

4. Alarm Function: The app includes an alarm feature to prevent users from forgetting to take their medication. Users can set customized reminders to ensure timely intake.

5. Family Management: The app allows users to manage not only their own medication records but also those of their family members, providing a convenient solution for household medication management.

#### Explanation of Functions:

**Medication Registration:** Users can add frequently used medications to their personalized medication list, eliminating the need to manually enter the medication name each time. By recording the number of days' worth of medication as specified in the prescription, users can set the alarm period in advance, streamlining the process of medication management.

**Medication Recording:** With a simple press of the record symbol and selection of the medication, users can easily keep track of the medications they have taken or used. In case of a missed record, users have the flexibility to choose the time to retroactively log their medication. The app conveniently organizes all medication records in a comprehensive list, facilitating an overview of the user's medication history.

#### App Description:

Let this app take care of your medication records. By recording the medications you have taken or used and the corresponding time, you can conveniently refer back to the app to verify if you have taken your medication when memory fails. The app also features an alarm function that sends reminders to prevent you from forgetting to take your medicine. This app is ideal for individuals who desire to keep a record of their medication intake while primarily seeking a reliable function to prevent medication forgetfulness.

- **Medisafe Pill & Med Reminder**

This app has garnered the recognition of pharmacists as the leading medication reminder app in the industry. With an impressive rating of 4.7 out of 5 stars from over 250,000 users on the app store, it has maintained its popularity for nearly a decade.



*Figure 2- 7*

Upon entering your medication information, the app generates a comprehensive report that outlines the specific timings for each medication. This report can be shared with a family member or caregiver, allowing them to keep track of your medication intake and identify any missed doses.

Medisafe goes beyond reminders and offers additional features to enhance medication management. It sends notifications when your medication supply is running low, reminding you to refill your prescriptions. Furthermore, it provides medication interaction warnings, alerting you to potential conflicts with certain foods or alcoholic beverages. For those who choose to do so, the app enables the synchronization of health data, such as blood pressure, heart rate, and blood sugar levels.

Highlighted Features:

- Pill reminder and alarm system catering to all medication needs.
- Drug-to-drug interaction checker for enhanced safety.
- Family and caregiver support through the "Medfriend" functionality.
- Medicine tracker to monitor medication intake.
- Refill reminders to ensure an adequate medication supply.
- Doctor appointment manager and calendar for effective healthcare planning.
- Support for complex dose schedules to accommodate individual requirements.
- Capability to add "as needed" medications, vitamins, and supplements.



- Extensive selection of over-the-counter (OTC) and prescription medications.
- Daily, weekly, and monthly medication reporting with a logbook for sharing with your doctor.
- Tracking of health measurements associated with various medical conditions, including diabetes, hypertension, cancer, anxiety, depression, HIV, multiple sclerosis (MS), Crohn's disease, lymphoma, myeloma, and leukemia.
- Android Wear compatibility for seamless integration with wearable devices.
- Customizable reminders and time settings, allowing for personalized scheduling preferences such as a weekend mode for more relaxed wake-uptimes.

## 2.2 Comparative study

<b>App Name</b>	<b>Best Feature</b>	<b>Pros</b>	<b>Cons</b>	<b>Rating</b>	<b>Cost</b>
MedsLog	Non	Medicine Reminder	Bad Interface Costly	2/5	4\$
MedHelper	Database stores information about each medicine	Friendly Interface Take Notes	Requires Internet Connection	4/5	Free
DoseCast	Multiple Devices	Friendly Interface Customize Ringtone	Costly Requires internet connection	4/5	\$5
Mango Health	Healthy Activity	Friendly Interface	Requires Internet Connection	3.5/5	Free
Every Dose	Virtual Assistant	Friendly Interface	Requires Internet Connection	4.5/5	Free
MyTherapy Pill Reminder	Link With Specific Doctor	Friendly Interface Customize Ringtone	Requires Internet Connection	4/5	Free
Medisafe Pill & Med Reminder	Group Sharing	Good User Interface	Working Online	3.5/5	Free

Table 2-1



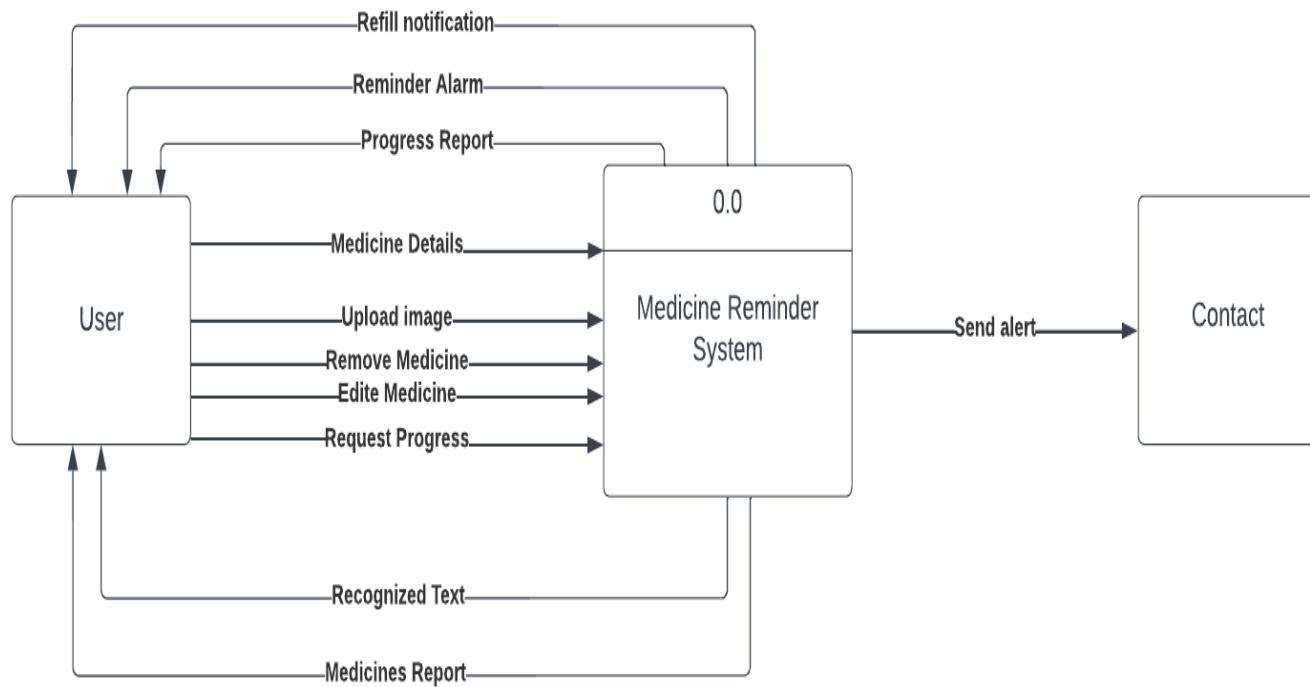
### **2.3 Expected outcomes:**

1. A fully functional medication reminder app
2. An alarm that reminds patients to take their medications on time.
3. Notifications that remind patients to refill their medications before they run out.
4. Export Medical Log
5. Text Recognition Feature

## Chapter Three

# 3 SYSTEM ANALYSIS AND DESIGN

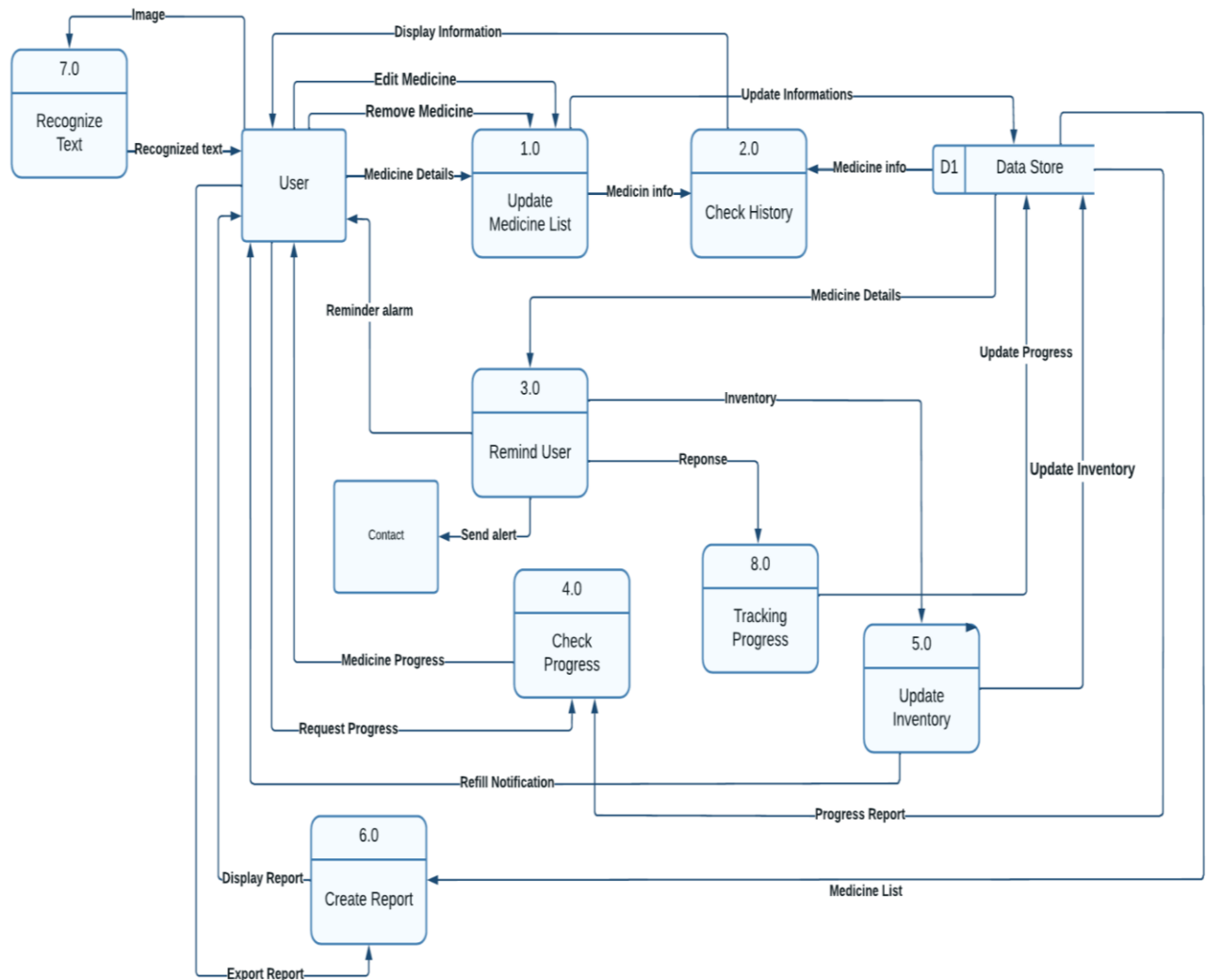
### 3.1 Context Diagram



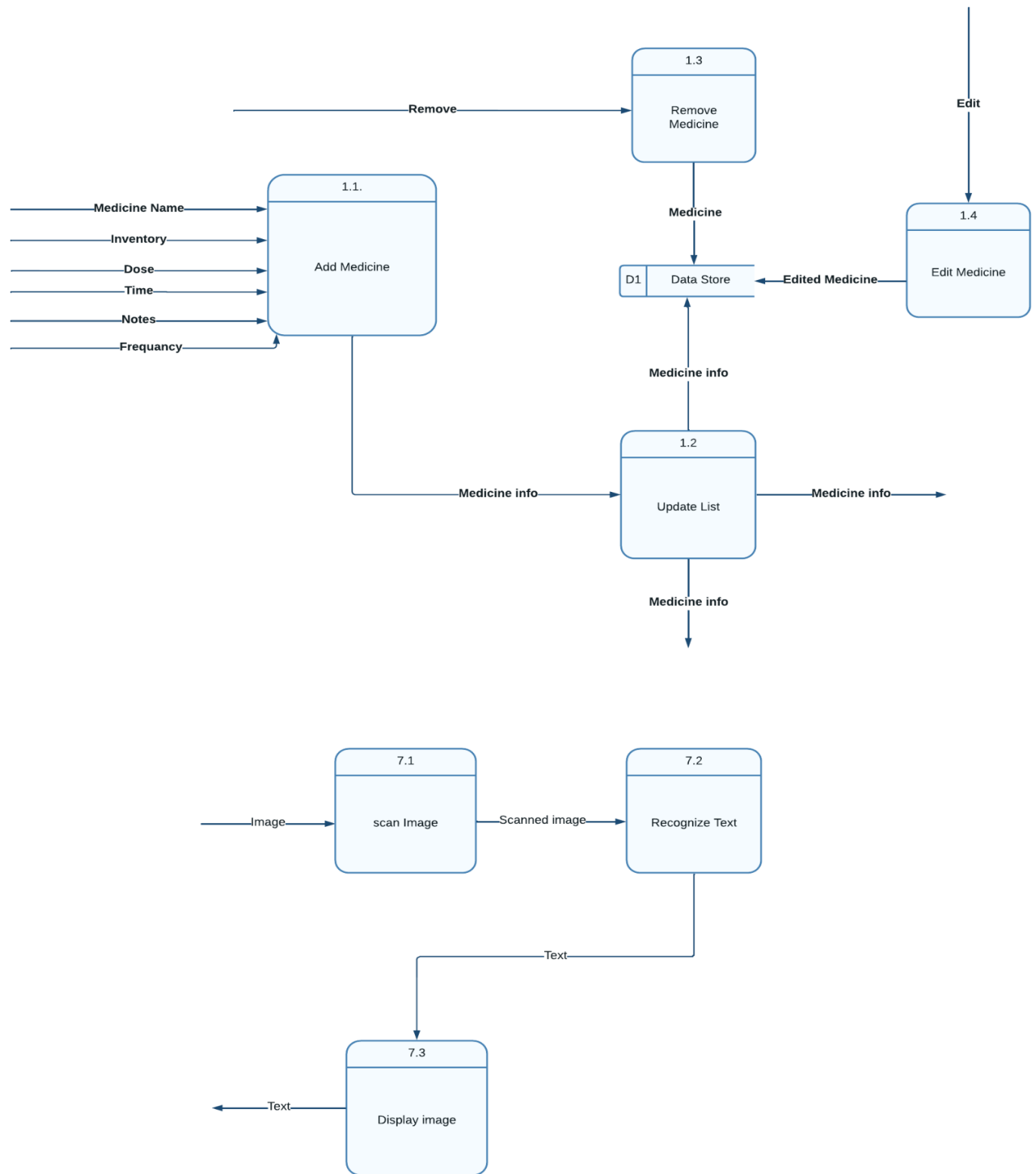


### 3.2 Data Flow Diagram

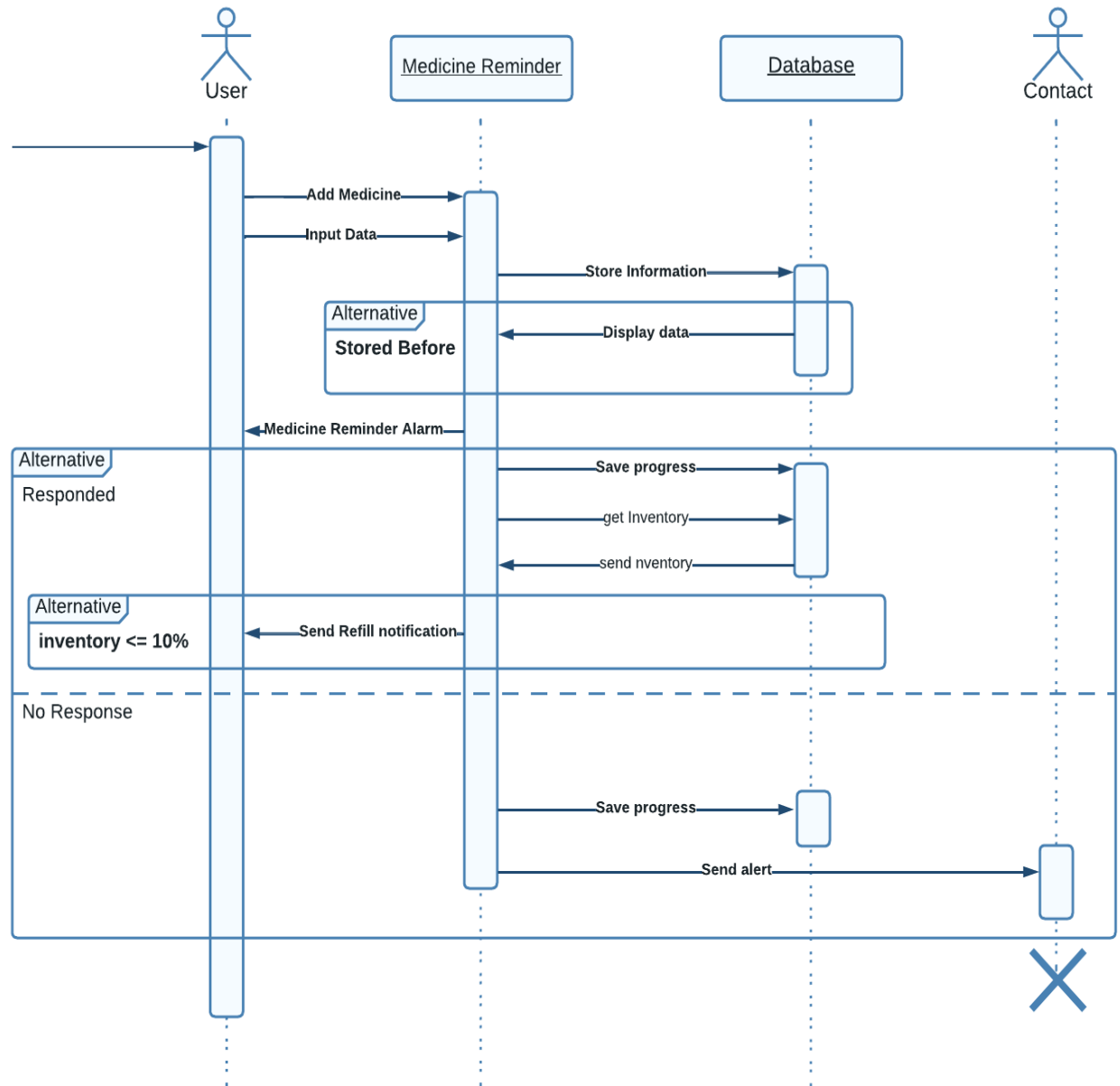
#### 3.2.1 Level 1

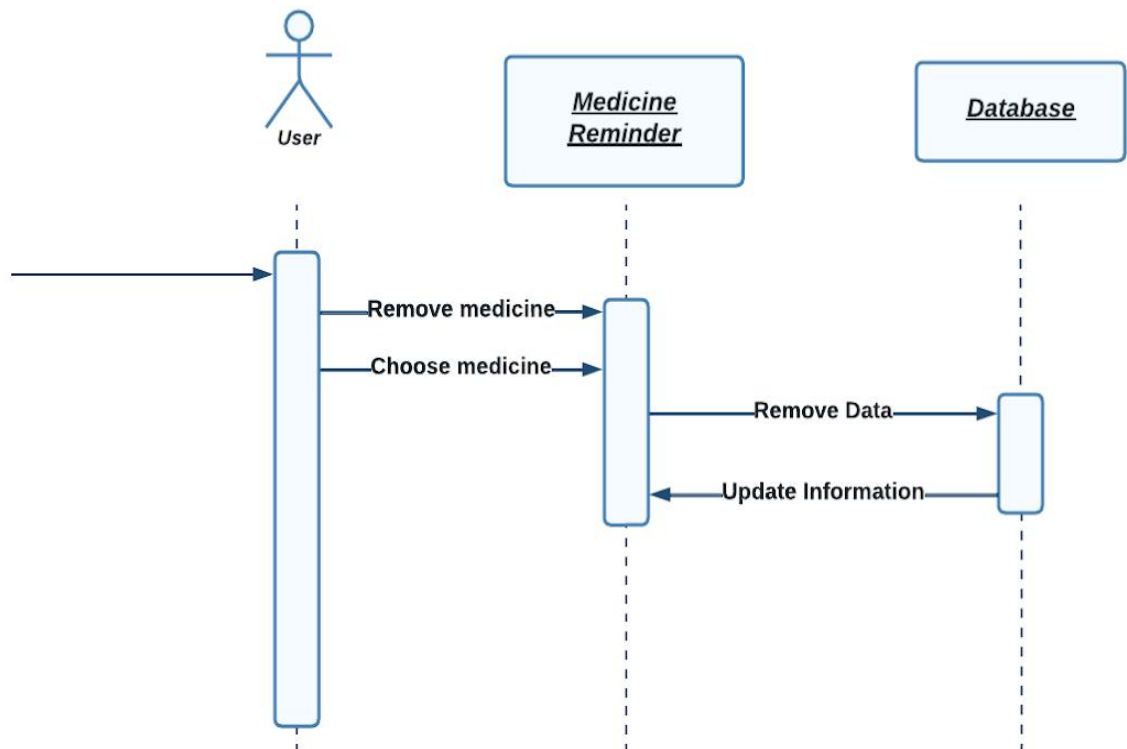
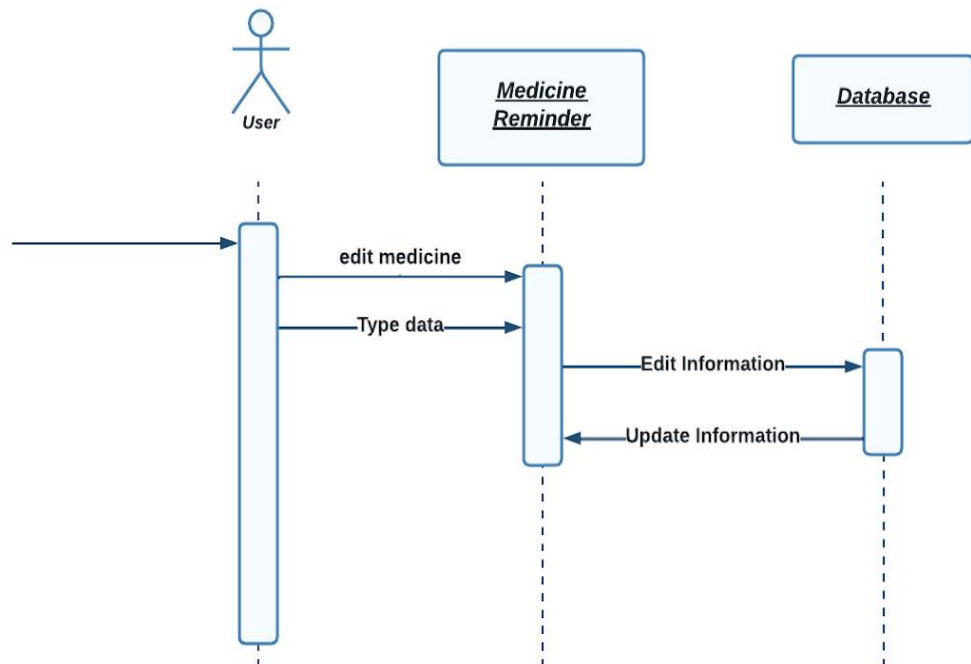


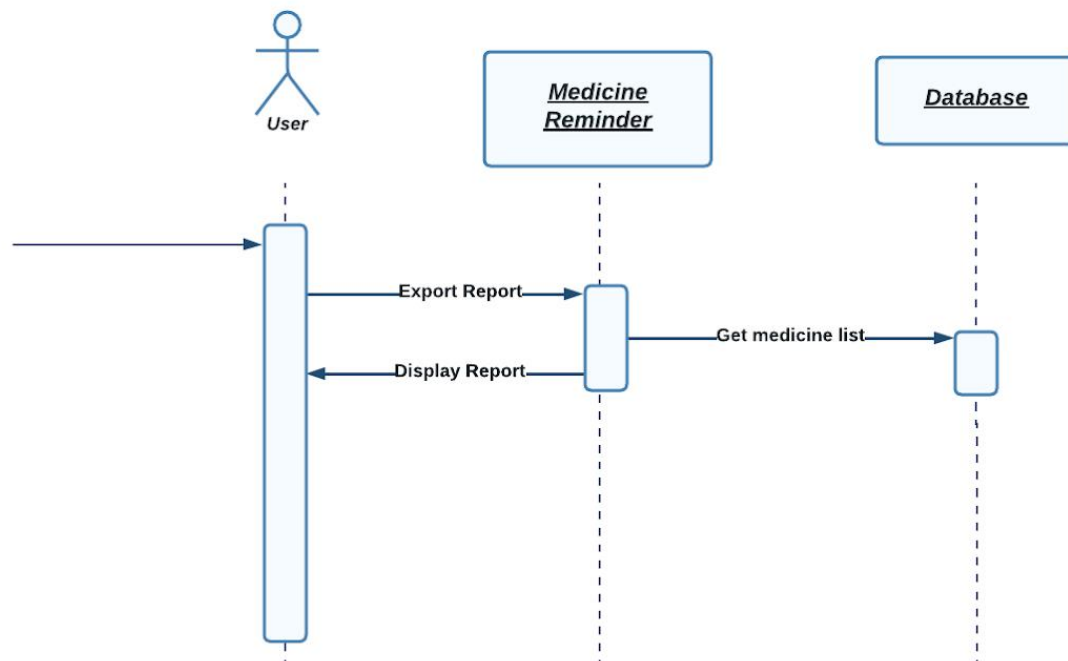
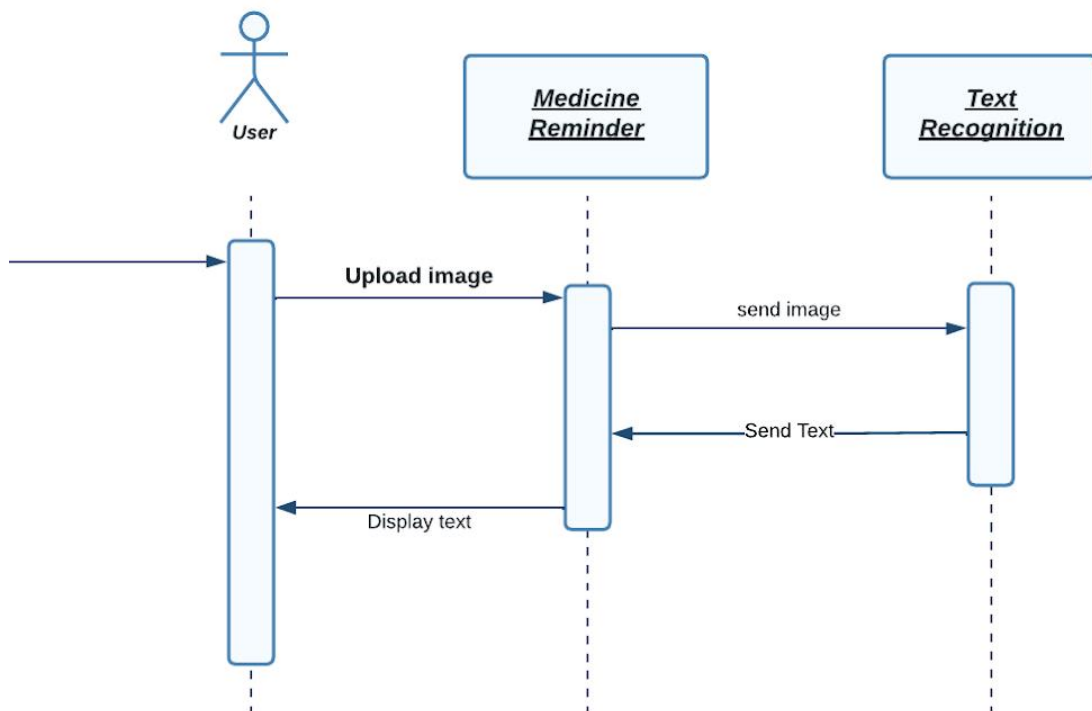
### 3.2.2 Level 2



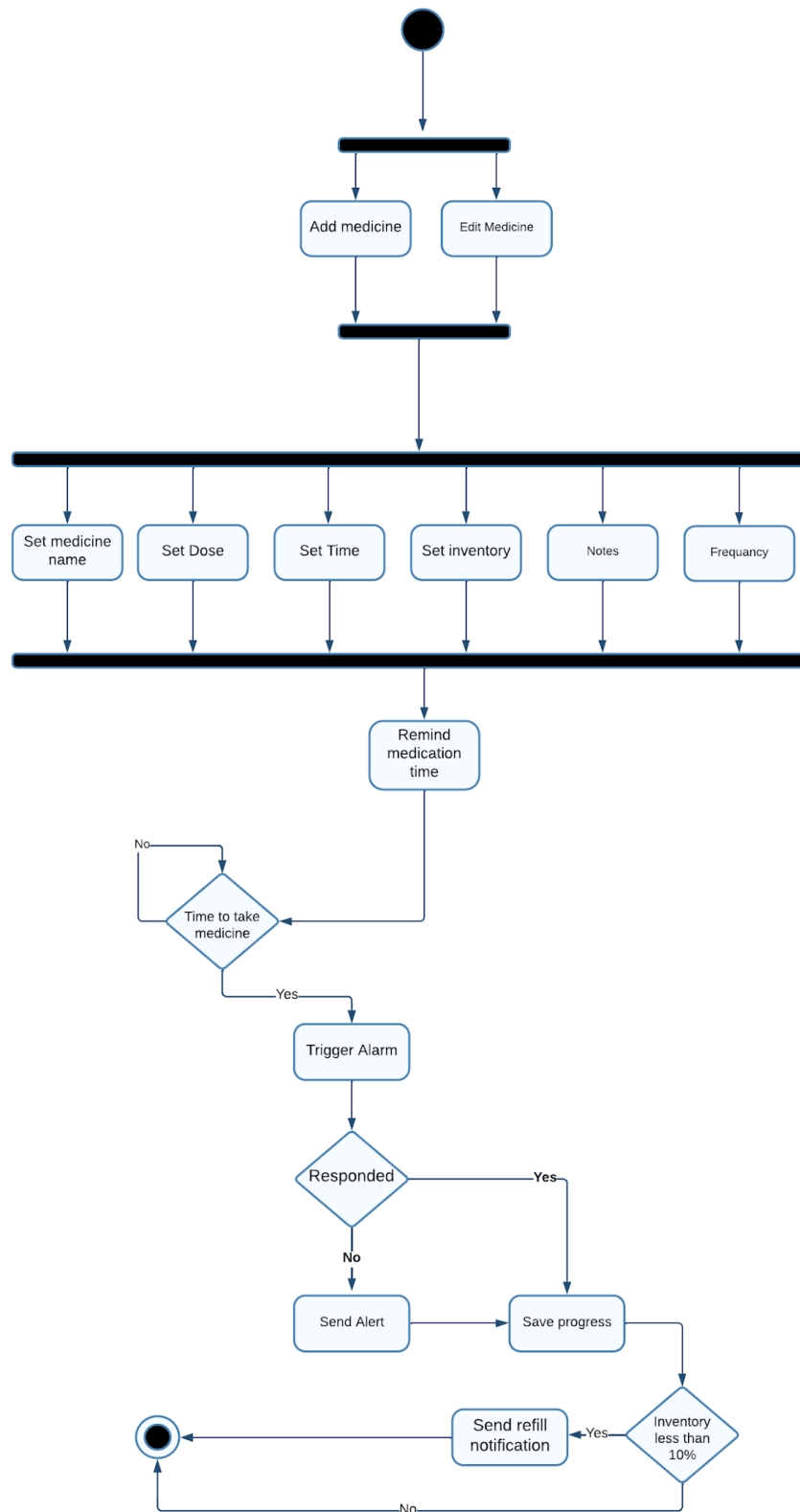
### 3.3 Sequence Diagram

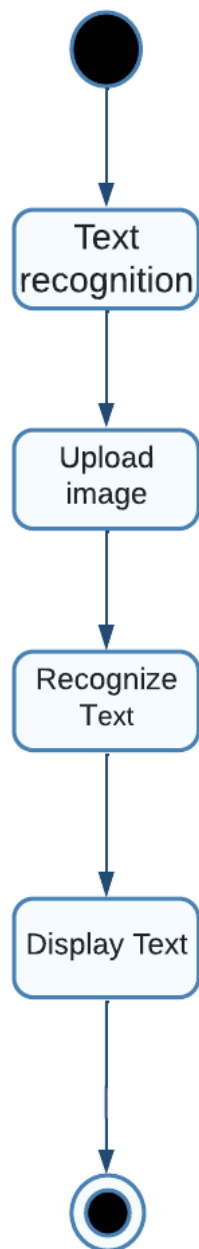




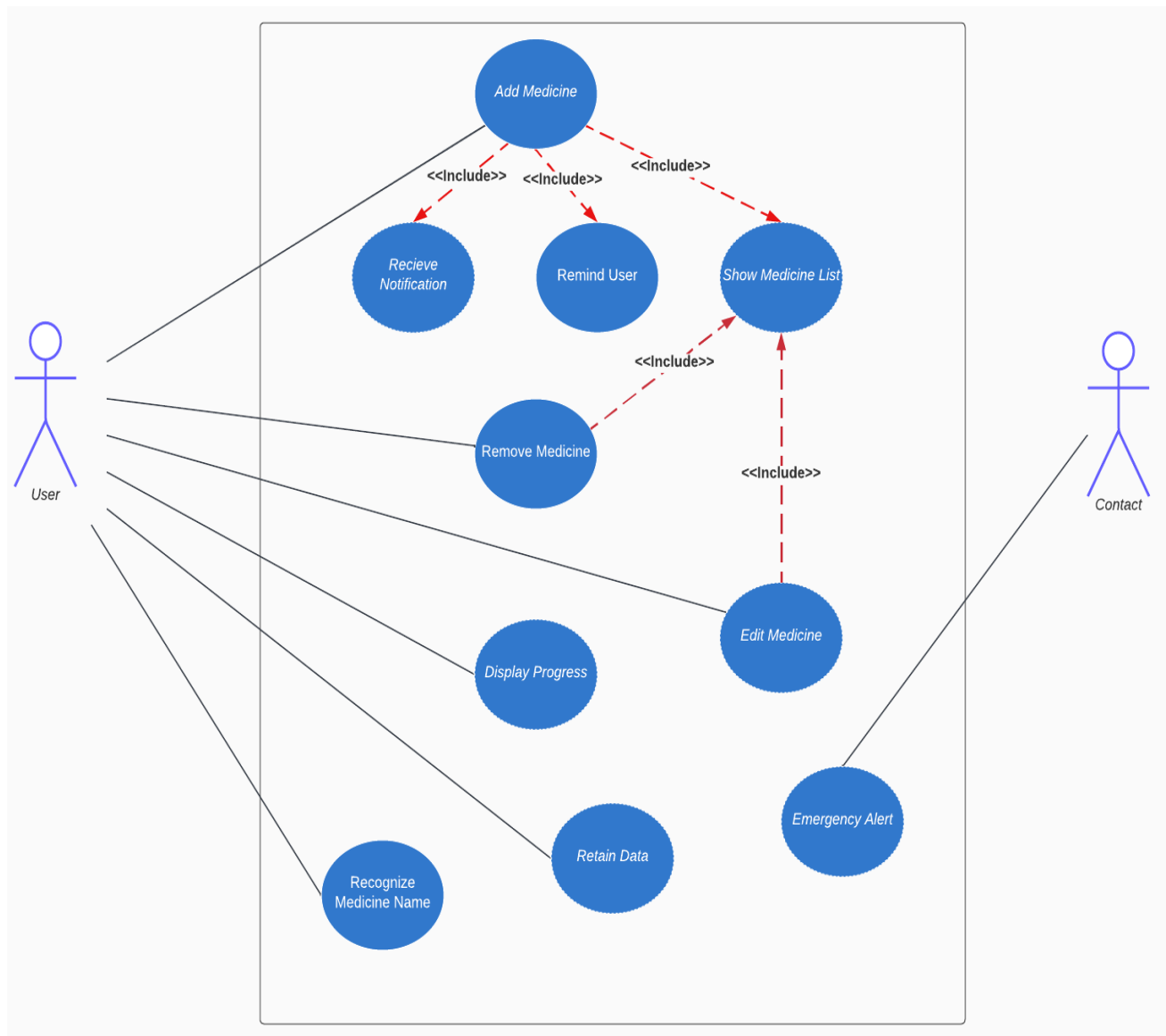


### 3.4 State Chart



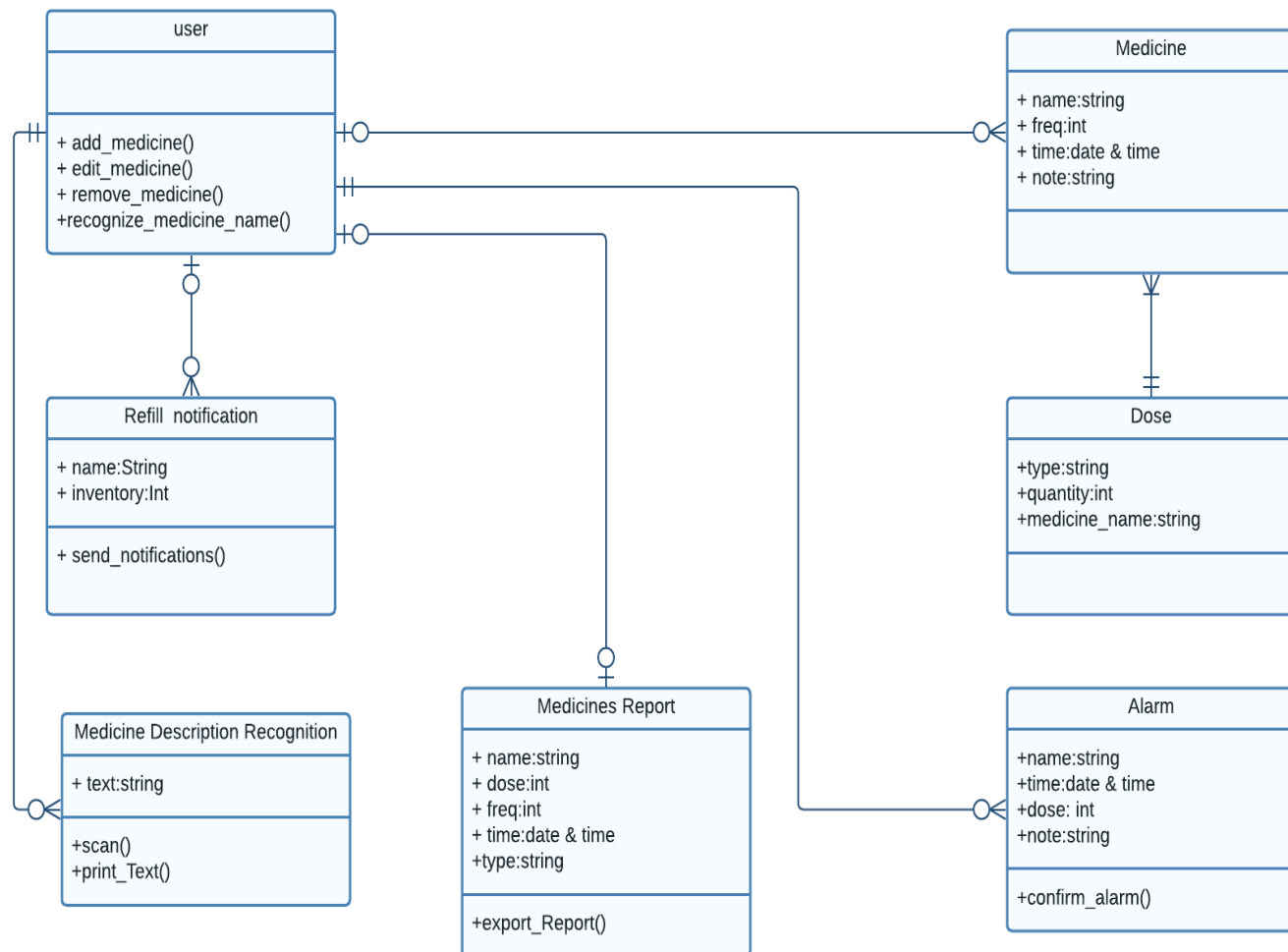


### 3.5 Use Case Diagram

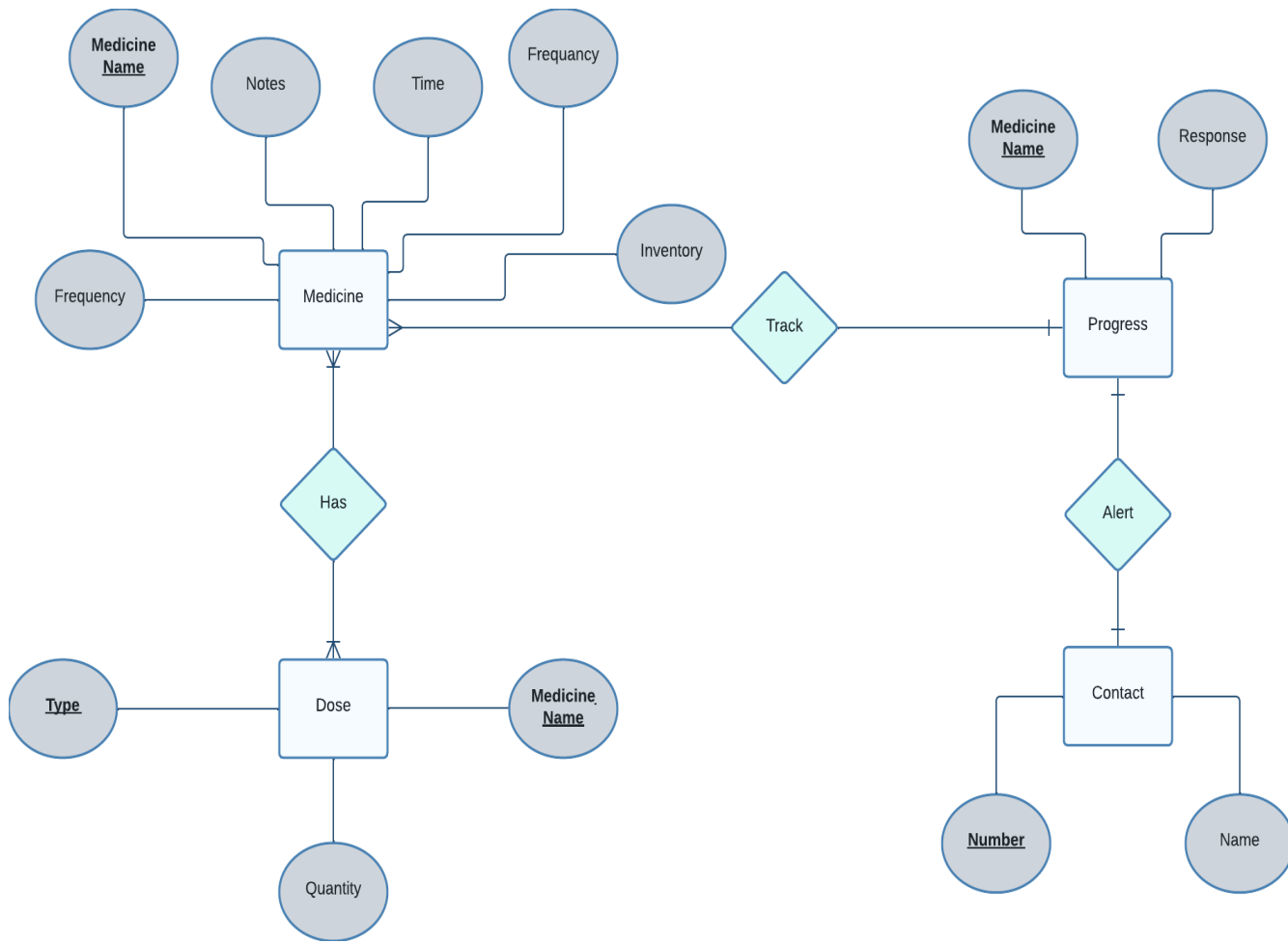




### 3.6 Class Diagram



### 3.7 Entity Relationship Diagram



## Chapter Four

# 4 PROPOSED SYSTEM

### 4.1 System Architecture

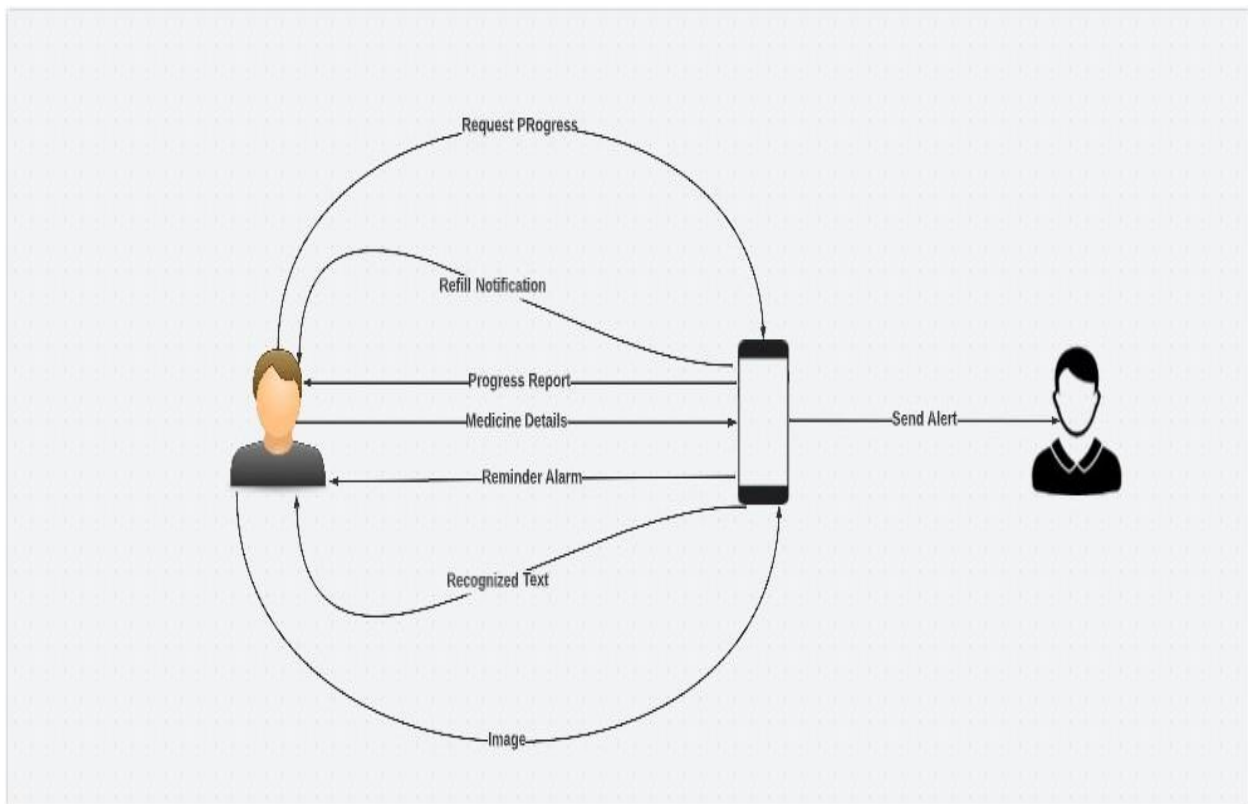


Figure 4- 1

This figure shows a basic idea of how the medicine tracker application would work the user will first open the app then he/she can use any feature form the app for example the text recognition which will prompt the user to take an image then display the text in this image on the screen. The user also can add medicine details and set a time to take that medicine and the app will remind the user to take it on time by sounding an alarm.

## 4.2 Algorithms and Methodologies

### Methodologies:

#### **Rapid application development (RAD)**

Rapid Application Development (RAD) methodology is an iterative and incremental approach to software development that emphasizes rapid prototyping and quick feedback cycles. It is designed to expedite the development process by focusing on delivering functional prototypes early on, allowing for continuous improvements based on user feedback.

At its core, RAD methodology aims to accelerate the development timeline while ensuring high-quality outcomes. It achieves this by involving users, developers, and stakeholders in a collaborative and iterative process. The methodology emphasizes flexibility, adaptability, and responsiveness to change, making it particularly suitable for projects where requirements may evolve or where time-to-market is crucial.

One of the key features of RAD methodology is the use of prototyping to quickly visualize and validate system functionalities. Instead of spending extensive time on detailed documentation, RAD encourages the development of functional prototypes that allow users and stakeholders to provide early feedback. This iterative feedback loop fosters continuous improvement and ensures that the final product aligns with the user's expectations.

RAD methodology promotes cross-functional teams and promotes effective communication and collaboration among team members. Developers, designers, business analysts, and end-users work closely together, fostering a shared understanding of project goals and requirements. This collaborative environment facilitates quick decision-making, efficient problem-solving, and a shared sense of ownership over the project's success.

Furthermore, RAD methodology emphasizes the use of reusable components and code libraries to accelerate development speed. By leveraging existing resources and building upon proven solutions, RAD teams can streamline the development process and reduce the time and effort required to deliver high-quality software.

In terms of project management, RAD methodology typically employs time-boxed iterations or sprints, with each iteration focused on delivering a specific set of features or functionality. These iterations are planned, executed, and evaluated in short cycles, allowing for rapid progress and frequent feedback. This iterative approach enables teams to adapt to changing requirements and pivot their development efforts accordingly.

While RAD methodology offers numerous benefits, it is important to consider potential challenges. For instance, the emphasis on speed and flexibility may increase the risk of overlooking certain aspects, such as comprehensive documentation or thorough testing. Therefore, proper quality assurance measures must be in place to ensure the reliability, security, and stability of the final product.

In conclusion, Rapid Application Development (RAD) methodology provides a dynamic and efficient approach to software development. By prioritizing rapid prototyping, iterative feedback, and collaborative teamwork, RAD enables teams to deliver functional software solutions quickly while ensuring continuous improvement based on user input. With its emphasis on flexibility and adaptability, RAD methodology is well-suited for projects that require rapid development, responsiveness to change, and a focus on user satisfaction. However, it is essential to balance speed with quality assurance measures to ensure the final product meets the required standards.

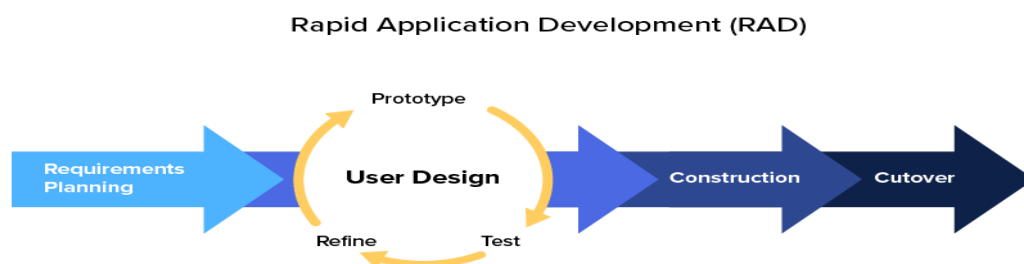


Figure 4- 2



## **Agile                      Software                      Development                      Methodology**

Agile Software Development methodology is a flexible and iterative approach to software development that focuses on delivering value to customers through continuous collaboration, adaptability, and rapid delivery of working software. It emphasizes close collaboration among cross-functional teams, customer involvement throughout the development process, and the ability to respond to change effectively.

At its core, Agile methodology is driven by the Agile Manifesto, which outlines a set of guiding principles. These principles prioritize individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. By embracing these principles, Agile teams strive to enhance customer satisfaction, foster efficient communication, and adapt to evolving requirements.

One of the fundamental aspects of Agile methodology is its iterative and incremental approach. Instead of a linear development process, Agile breaks the project into small iterations or time-bound sprints. Each iteration focuses on delivering a working increment of the software that provides value to the customer. The feedback received from customers and stakeholders during these iterations is used to inform subsequent development cycles, enabling the project to evolve and adapt based on real-time insights.

Agile methodology encourages cross-functional teams that consist of individuals with diverse skill sets, including developers, testers, designers, and business analysts. The collaborative nature of Agile teams promotes effective communication, shared responsibility, and collective decision-making. By fostering a collaborative environment, Agile methodology enhances team synergy and promotes a sense of ownership and accountability.

To facilitate collaboration and transparency, Agile teams often utilize visual boards or online tools to track progress and manage tasks. These tools, such as Kanban boards or Scrum boards, provide a shared view of the project's status, enabling team members to coordinate efforts, identify bottlenecks, and prioritize work effectively.



Continuous integration and frequent delivery of working software are also key principles of Agile methodology. Through practices like automated testing and continuous integration, Agile teams ensure that the software is thoroughly tested and functional at all times. This enables regular delivery of incremental software versions, allowing customers to provide feedback and steer the project's direction.

Another crucial aspect of Agile methodology is the concept of "emergent requirements" or the ability to adapt to changing customer needs. Rather than relying on rigid upfront planning, Agile teams embrace change as a natural part of the development process. Requirements are continuously refined and reprioritized based on customer feedback, allowing the team to deliver a product that meets the evolving needs of the customer.

While Agile methodology offers numerous benefits, it is important to consider potential challenges. For instance, the rapid pace of development and the need for continuous customer involvement may require significant time and effort from stakeholders. Additionally, maintaining a balance between flexibility and project discipline can be a challenge, as too much flexibility can lead to scope creep or lack of focus.

In conclusion, Agile Software Development methodology provides a flexible and collaborative approach to software development. By prioritizing customer collaboration, iterative development, and adaptability, Agile teams can deliver high-quality software that meets customer needs. Through its emphasis on continuous improvement, transparency, and rapid delivery, Agile methodology enables teams to respond to change effectively and deliver value early and consistently. However, it is crucial to strike a balance between flexibility and discipline to ensure successful project outcomes.



## **Algorithms:**

- **Text Recognition**

Text Recognition is an algorithm that converts typed or handwritten text and printed images containing text into machine-readable digital data format.

- **Inventory refill**

Inventory refill is an algorithm that reminds the user to refill the medication or buy another package if the package has less than 10% remaining.

- **Reminder**

The algorithm involves setting a specific time in the application, monitoring the current time, and comparing the two. When the current time matches the set time, the algorithm triggers the alarm sound and sends a notification message to the user.

- **Export**

This algorithm allows the user to export a list of all medicines saved on the application. The user can export the list to a pdf and can print it so it makes buying the medicine easier.





### 4.3 Proposed Software and Hardware

- Software Requirements:
  - Dart Programming Language
  - Flutter framework used to produce android applications that work on android smartphones
  - Operating System: Windows (10/11) - Android (Pie 9.0)
  - IDE: Visual Studio Code / Android Studio
- Hardware Requirements:
  - Processor: intel core i7 11th generation
  - Ram: 16 GigaByte (GB)
  - Hard-Disk: 1 TeraByte (TB)
  - Any android device that can run android application is a suitable hardware for the Medicine Tracker project



## **4.4 Functional requirements**

- Manual entry of Medicine Details
- List of Medicines to be taken
- Reminder
- Text Recognition
- Export Report
- Check Progress
- Refill Reminder
- Emergency alert

## **4.5 Nonfunctional requirements**

- Usability
- Efficiency
- Reliability
- Availability
- Maintainability
- Performance

## *Chapter Five*

# **5 RESULTS AND DISCUSSION**

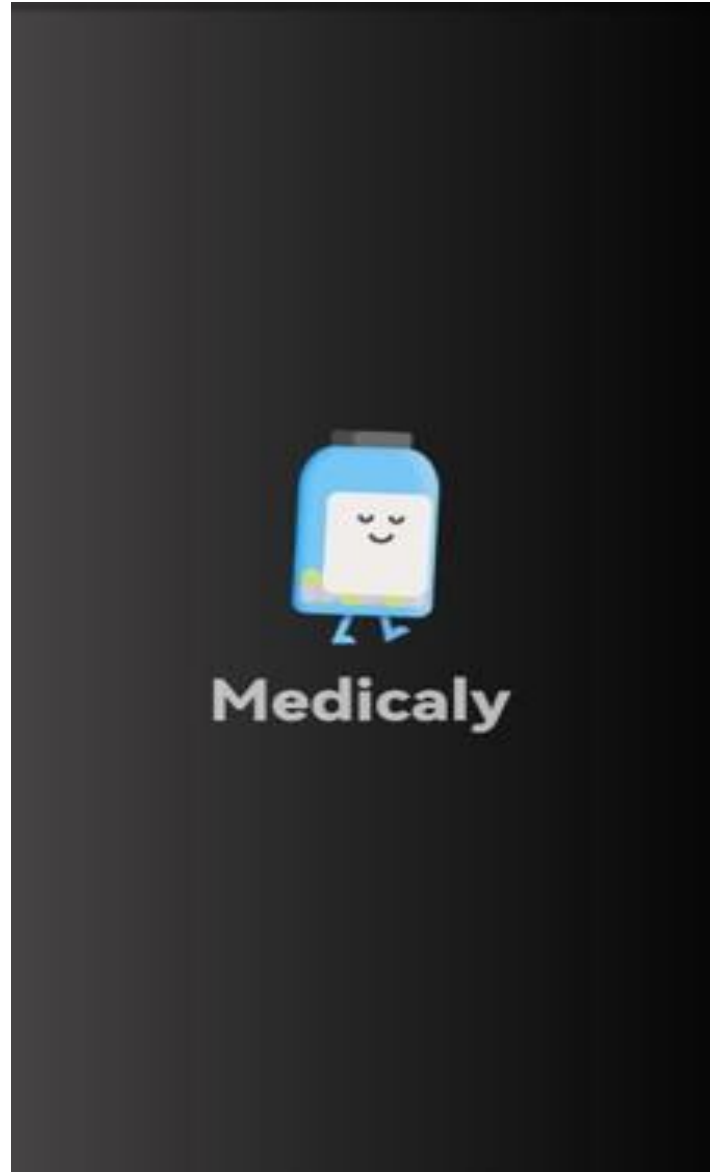
### **5.1 Application Results**

The medicine reminder app was meticulously designed to employ the power of push notifications, ensuring timely medication reminders for patients at their prescribed intervals. With its intuitive interface, the app empowered patients to effortlessly track their medication intake while offering the flexibility to configure personalized reminders tailored to their specific medication regimens.

The outcomes of our comprehensive study helps on the remarkable potential of the medicine reminder app in bolstering medication adherence, particularly among individuals grappling with chronic illnesses. The app's multifaceted features, such as push notifications and customizable reminders, emerged as pivotal tools in fostering medication adherence and curbing the occurrence of missed doses. These findings hold immense significance, considering the alarming rates of non-adherence observed among patients managing chronic illnesses. By mitigating the challenges associated with non-adherence, the app has the potential to catalyze improvements in health outcomes while simultaneously reducing the burden on healthcare systems.

Our study serves as a compelling testament to the effectiveness of the medicine reminder app as a valuable tool in enhancing medication adherence among patients with chronic illnesses. As we forge ahead, it is essential to embark on further investigations to explore the app's long-term effects on medication adherence, while simultaneously evaluating its efficacy in diverse healthcare settings and among varied patient populations. Through rigorous research endeavors, we can unearth a wealth of insights that will continue to refine and optimize the app's functionality, ultimately revolutionizing medication management practices and fostering improved health outcomes for individuals worldwide.

This figure shows the loading screen of the medicine tracker application when you press on the app icon.



*Figure 5- 1*

This figure shows the home page of the medicine tracker application where all the medicines the user added are shown. From this page the user also can check the information of each medicine such as time, quantity, medicine name, type of medicine and frequency.

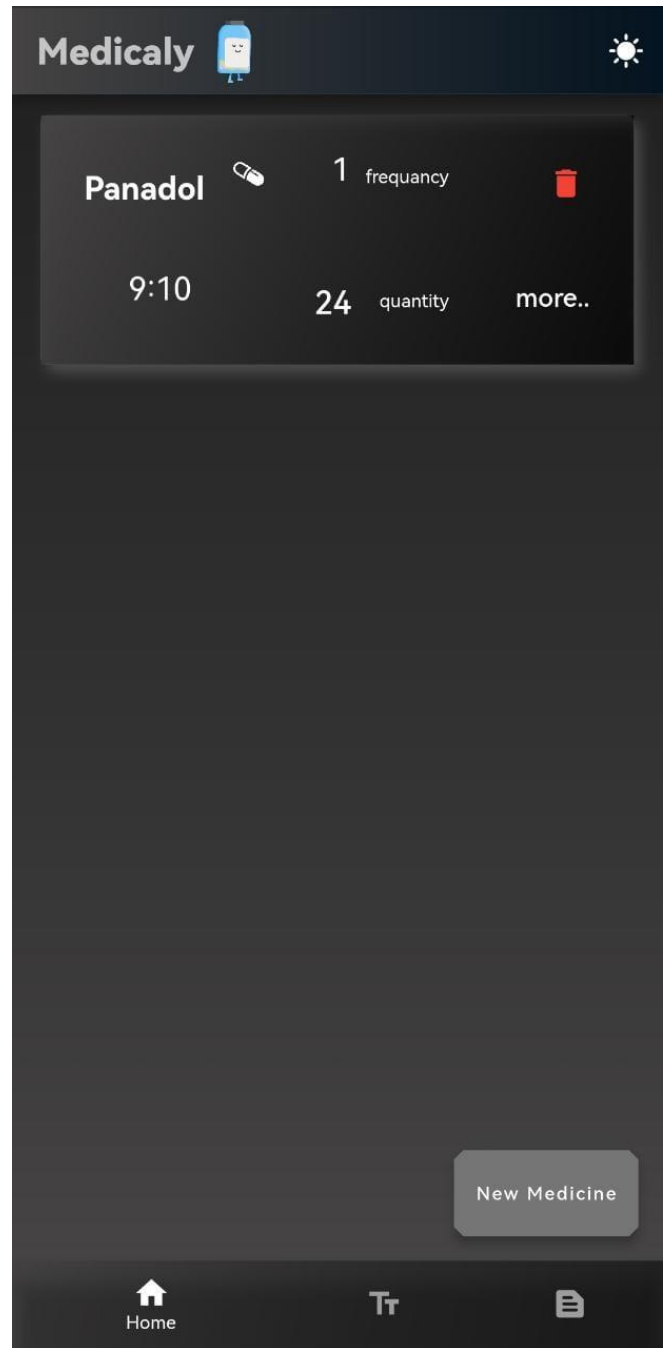
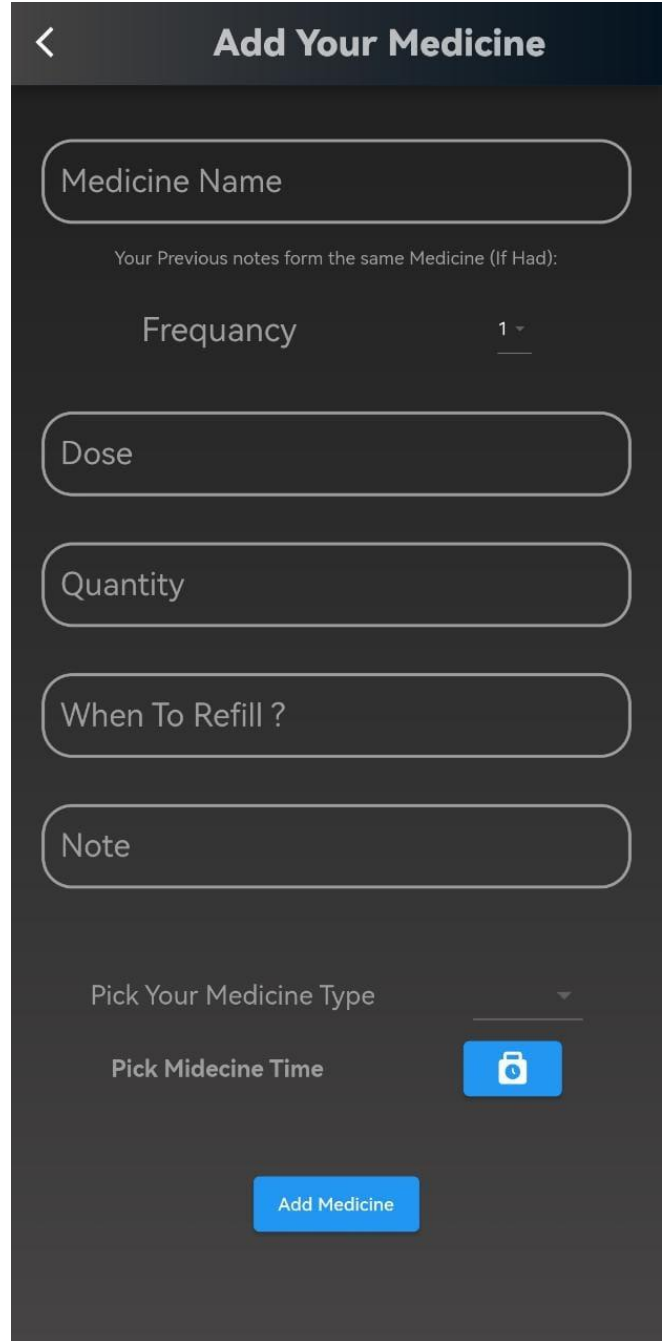


Figure 5- 2

This figure shows the add medicine page of the medicine tracker application where the user adds all the information about the medicine they want to be reminded of taking some of these information is medicine name, dose, note, quantity and time.

The image shows a mobile application screen titled "Add Your Medicine". At the top left is a back arrow icon. Below the title are several input fields: "Medicine Name", "Frequency" (with a dropdown menu showing "1"), "Dose", "Quantity", "When To Refill ?", and "Note". Below these fields are two more dropdown menus: "Pick Your Medicine Type" and "Pick Midicine Time" (with a blue button featuring a white pill icon). At the bottom center is a blue button labeled "Add Medicine".

< **Add Your Medicine**

Medicine Name

Your Previous notes form the same Medicine (If Had):

Frequency 1

Dose

Quantity

When To Refill ?

Note

Pick Your Medicine Type

Pick Midicine Time

Add Medicine

Figure 5- 3

This figure shows the text recognition page of the medicine tracker application in this page the user can take a picture of the medication description that comes in the package and then press on generate text which then displays on the screen the text written in a much larger font allowing users who have difficulty reading small font to be more comfortable also there is a feature that allows the user to translate the displayed text into Arabic.

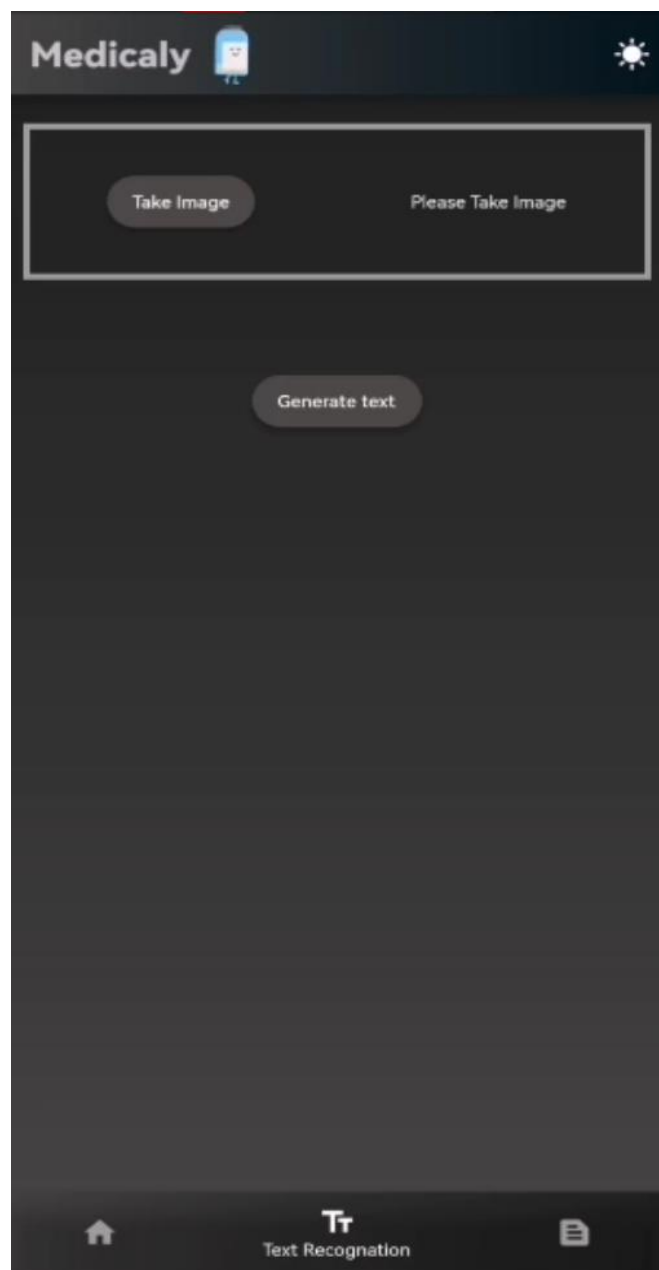


Figure 5- 4

This figure shows the export medicine list page of the medicine tracker application. In this page the user can easily export all the medicine he/she takes into a pdf which then could be printed so they can have a soft and a hard copy of the medicine list which will make the process of buying the medicine easier for them.

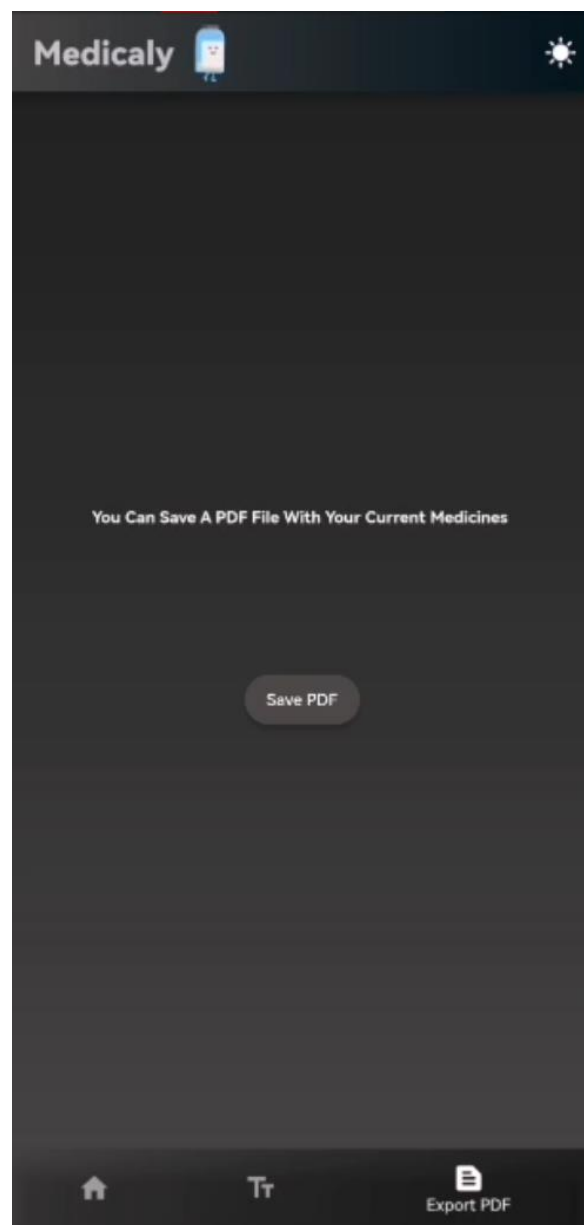


Figure 5- 5



### **Conclusion:**

In conclusion, the medicine tracker mobile application emerges as an invaluable asset for individuals seeking to manage their medication schedules with utmost efficacy. Equipped with a diverse range of features, including medication reminders, dosage tracking, and prescription management, this application revolutionizes the way users engage with their medication regimens. By cultivating a user-friendly interface complemented by extensive customization options, the app empowers users to effortlessly monitor their medication intake, ensuring that they adhere to the prescribed dosages at the designated times.

Beyond its fundamental functionalities, the medicine tracker application serves as a comprehensive repository, allowing users to maintain a detailed record of their medication history. This invaluable feature facilitates seamless communication with healthcare providers, enabling users to articulate their progress accurately and effectively. By harnessing the app's capabilities, individuals can take control of their health journey, empowering themselves to make informed decisions and achieve optimal outcomes.

In essence, the medicine tracker mobile application stands as an indispensable tool for individuals across diverse backgrounds who seek to manage their medication regimens with precision and efficacy. Through its seamless integration into users' daily lives, this application becomes a steadfast companion, empowering individuals to assume an active role in their healthcare. By leveraging the app's comprehensive functionalities, users can embark on a path toward improved health outcomes, enabling them to lead healthier and more fulfilling lives.



## **References**

1. MEDHELPER INC. BY MONTREAL BASED HEALTHCARE TECHNOLOGY START-UP  
<https://medhelper.com/medhelper/>
2. MEDSLOG BY MODESITT SOFTWARE  
<https://www.nytimes.com/2010/09/30/technology/personaltech/30smart.html>
3. DOSECAST BY MONTUNO SOFTWARE, LLC  
<https://www.montunosoftware.com/about/>  
<https://apps.apple.com/us/app/dosecast-my-pill-reminder-app/id365191644>
4. <https://www.studocu.com/in/document/institute-of-management-and-research-pune/operation-management/medicine-reminder-android-app-project-report/22430306>
5. CUTE PILL BY FUTASAJI LCC  
<https://play.google.com/store/apps/details?id=net.futasaji.medicine&hl=en&gl=US>
6. EVERYDOSE BY GROOVE HEALTH INC, <https://www.everydose.ai/>
7. MEDISAFE BY TRUEPILL <https://www.medisafeapp.com/>
8. MANGO HEALTH BY TRIALCARD <https://support.mangohealth.com/hc/en-us>



## APPENDIX A: CODE USED FOR DEVELOPMENT

### Add medicine page:

```
import 'package:awesome_notifications/awesome_notifications.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:medically/widgets.dart';
import 'homePage.dart';
import 'notification.dart';
import 'sql.dart';
```

```
class addMedicinePage extends StatefulWidget {
  @override
  State<StatefulWidget> createState() => _addMedicinePageState();
}
```

```
int freqNumber = 0;
```

```
class _addMedicinePageState extends State<addMedicinePage> {
  TextEditingController nameController = TextEditingController();
  TextEditingController freqController = TextEditingController();
  TextEditingController timeController = TextEditingController();
  TextEditingController noteController = TextEditingController();
  TextEditingController quantityController = TextEditingController();
  TextEditingController doseController = TextEditingController();
  TextEditingController refillController = TextEditingController();
  TextEditingController timesController = TextEditingController();
  TimeOfDay selectedTime = TimeOfDay.now();
  String timeString = "";
  String timeString2 = "";
  List<String> freqList = ['1', '2'];
  List<String> type = ['pill', 'syrup', 'injection', 'drop'];
  SqlDb sqldb = SqlDb();
  bool pill = false;
  int Pill = 0;
  String typeVal = 'select';
  List<Map>? notes;
  List<String> matchingMedicine = [];
```

```
_selectTime(BuildContext context, String t) async {
  final TimeOfDay? timeOfDay = await showTimePicker(
    context: context,
    initialTime: selectedTime,
    initialEntryMode: TimePickerEntryMode.dial,
  );
  if (timeOfDay != null && timeOfDay != selectedTime) {
    setState() {
      selectedTime = timeOfDay;
    }
  }
}
```



```

        timeString =
            selectedTime.hour.toString() + ':' + selectedTime.minute.toString();
    });
}
}

selectTime2(BuildContext context, String t) async {
  final TimeOfDay? timeOfDay = await showTimePicker(
    context: context,
    initialTime: selectedTime,
    initialEntryMode: TimePickerEntryMode.dial,
  );
  if (timeOfDay != null && timeOfDay != selectedTime) {
    setState(() {
      selectedTime = timeOfDay;
      timeString2 =
        selectedTime.hour.toString() + ':' + selectedTime.minute.toString();
    });
  }
}

@override
Widget build(BuildContext context) {

  read() async {
    List<Map> response = await sqlDb.read('SELECT * FROM notes');

    notes = response;
    print(notes);
  }

  SqlDb sql = SqlDb();
  Future<List<Map>> readData() async {
    List<Map> response = await sql.read('SELECT * FROM medicine');
    setState(() {
      medList = response;
    });
    return response;
  }

  return Scaffold(
    appBar: AppBar(
      flexibleSpace: Dark == true
        ? Container(
            decoration: const BoxDecoration(
              gradient: LinearGradient(colors: [
                Color.fromARGB(255, 70, 68, 68),
                Colors.black87
              ])))
        : null,
    elevation: 6,

```



```

leading: IconButton(
  onPressed: () {
    Navigator.of(context).pushReplacementNamed('/home');
  },
  icon: Icon(Icons.arrow_back_ios)),
title: Row(
  mainAxisAlignment: MainAxisAlignment.center,
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Text(
      'Add Your Medicine',
      style: TextStyle(
        fontSize: 28,
        fontWeight: FontWeight.w900,
        color: Colors.white70),
    ),
  ],
),
),
body: Container(
  height: double.infinity,
  decoration: Dark == true
    ? BoxDecoration(
      gradient: LinearGradient(
        colors: [Color.fromARGB(255, 70, 68, 68), Colors.black87],
        begin: Alignment.bottomCenter,
        end: Alignment.topCenter))
    : null,
child: ListView(children: [
  Padding(
    padding: const EdgeInsets.all(20),
    child: Column(
      children: [
        Container(
          margin: EdgeInsets.symmetric(vertical: 15),
          child: TextField(
            controller: nameController,
            decoration: const InputDecoration(
              enabledBorder: OutlineInputBorder(
                borderSide: BorderSide(width: 2, color: Colors.grey),
                borderRadius: BorderRadius.all(Radius.circular(20))),
              label: Text('Medicine Name',
                style: TextStyle(color: Colors.grey, fontSize: 24)),
              hintText: 'Enter Medicine Name',
              hintStyle: TextStyle(color: Colors.grey, fontSize: 10),
            ),
            style: const TextStyle(color: Colors.grey, fontSize: 15),
            onChanged: (value) {
              setState(() {
                matchingMedicine = [];
              });
            }
          ),

```



```

read();
if (notes != null) {
  for (int note = 0; note < notes!.length; note++) {
    if (notes![note]['medicine'] == value) {
      setState(() {
        matchingMedicine.add(notes![note]['note']);
      });
    } else if (notes![note]['medicine'] != value) {
      continue;
    }
  }
}

print(matchingMedicine);
},
),
),
matchingMedicine != []
? Center(
  child: Column(
    children: [
      Text(
        'Your Previous notes form the same Medicine (If Had):',
        style: TextStyle(color: Colors.grey),
      ),
      for (var i in matchingMedicine)
      Text(
        i,
        style:
          TextStyle(fontSize: 15, color: Colors.grey),
        textAlign: TextAlign.start,
      ),
    ],
  ))
: Container(),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    Container(
      child: Text(
        'Frequency',
        style: TextStyle(fontSize: 25, color: Colors.grey),
      ),
    ),
    Container(
      margin: EdgeInsets.symmetric(vertical: 15),
      child: DropdownButton(
        dropdownColor: Dark
          ? Color.fromARGB(255, 77, 73, 73)
          : Colors.white,
        style: TextStyle(

```



```

        color: Dark ? Colors.white : Colors.black),
        iconSize: 15,
        value: freqNumber >= 1 ? freqNumber.toString() : null,
        items: freqList.map((String items) {
          return DropdownMenuItem(
            value: items,
            child: Text(items),
          );
        }).toList(),
        onChanged: (String? value) {
          setState(() {
            freqNumber = int.parse(value!);
            freqController.text = value;
          });
        },
      ),
    ],
  ),
  Container(
    margin: EdgeInsets.symmetric(vertical: 15),
    child: TextField(
      controller: doseController,
      keyboardType: TextInputType.number,
      decoration: const InputDecoration(
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(width: 2, color: Colors.grey),
          borderRadius: BorderRadius.all(Radius.circular(20))),
        label: Text('Dose',
          style: TextStyle(color: Colors.grey, fontSize: 24)),
        hintText: 'Enter Medicine Dose you will take',
        hintStyle: TextStyle(color: Colors.grey, fontSize: 10),
      ),
      style: const TextStyle(color: Colors.grey, fontSize: 15),
    ),
  ),
  Container(
    margin: EdgeInsets.symmetric(vertical: 15),
    child: TextField(
      controller: quantityController,
      keyboardType: TextInputType.number,
      decoration: const InputDecoration(
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(width: 2, color: Colors.grey),
          borderRadius: BorderRadius.all(Radius.circular(20))),
        label: Text('Quantity',
          style: TextStyle(color: Colors.grey, fontSize: 24)),
        hintText: 'Enter Medicine Quantity You Have',
        hintStyle: TextStyle(color: Colors.grey, fontSize: 10),
      ),
      style: const TextStyle(color: Colors.grey, fontSize: 15),
    ),
  ),

```



```

    ),
  ),
  Container(
    margin: EdgeInsets.symmetric(vertical: 15),
    child: TextField(
      controller: refillController,
      keyboardType: TextInputType.number,
      decoration: const InputDecoration(
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(width: 2, color: Colors.grey),
          borderRadius: BorderRadius.all(Radius.circular(20))),
        label: Text('When To Refill ?',
          style: TextStyle(color: Colors.grey, fontSize: 24)),
        hintText: 'Remind To Refill When Quantity Becoms',
        hintStyle: TextStyle(color: Colors.grey, fontSize: 10),
      ),
      style: const TextStyle(color: Colors.grey, fontSize: 15),
    ),
  ),
  Container(
    margin: EdgeInsets.symmetric(vertical: 15),
    child: TextField(
      controller: noteController,
      decoration: const InputDecoration(
        enabledBorder: OutlineInputBorder(
          borderSide: BorderSide(width: 2, color: Colors.grey),
          borderRadius: BorderRadius.all(Radius.circular(20))),
        label: Text('Note',
          style: TextStyle(color: Colors.grey, fontSize: 24)),
        hintText: 'Enter MEDicine Notes',
        hintStyle: TextStyle(color: Colors.grey, fontSize: 10),
      ),
      style: const TextStyle(color: Colors.grey, fontSize: 15),
    ),
  ),
  SizedBox(height: 30),
  Row(
    mainAxisAlignment: MainAxisAlignment.spaceAround,
    children: [
      Text(
        'Pick Your Medicine Type',
        style: TextStyle(fontSize: 20, color: Colors.grey),
      ),
      DropdownButton(
        style: TextStyle(
          color: Dark ? Colors.white : Colors.black),
        dropdownColor:
          Dark ? Color.fromARGB(255, 77, 73, 73) : null,
        items: type.map((String items) {
          return DropdownMenuItem(
            value: items,

```





```

        child: Text(items),
      );
    }).toList(),
    onChanged: (value) {
      if (value == null) {}
      print(value);
      switch (value) {
        case 'pill':
          {
            setState() {
              Pill = 1;
            };
          }
          break;

        case 'syrup':
          {
            setState() {
              Pill = 2;
            };
          }
          break;

        case 'injection':
          {
            setState() {
              Pill = 3;
            };
          }
          break;
        case 'drop':
          {
            setState() {
              Pill = 4;
            };
          }
          break;
      }
      print(Pill);
    })

  ],
),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceAround,
  children: [
    const Text(
      'Pick Medicine Time',
      style: TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.w600,

```



```

        color: Colors.grey),
    ),
    ElevatedButton(
      onPressed: () {
        _selectTime(context, timeString);
      },
      child: const Icon(Icons.punch_clock_rounded))
    ],
  ),
  (freqNumber == 2)
    ? Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: [
        const Text(
          'The Second One',
          style: TextStyle(
            fontSize: 18,
            fontWeight: FontWeight.w600,
            color: Colors.grey),
        ),
        ElevatedButton(
          onPressed: () {
            _selectTime2(context, timeString2);
          },
          child: const Icon(Icons.punch_clock_rounded))
      ],
    )
    : Row(),

  //Text(timeString),
  Container(
    margin: EdgeInsets.only(top: 40),
    child: ElevatedButton(
      child: Text('Add Medicine'),
      onPressed: () async {
        if (nameController.text.isNotEmpty &&
            freqController.text.isNotEmpty &&
            quantityController.text.isNotEmpty &&
            noteController.text.isNotEmpty &&
            timeString != " &&
            timeString2 != " &&
            doseController.text.isNotEmpty &&
            refillController.text.isNotEmpty &&
            freqNumber == 2) {
          int r = await sqldb.insert(
            'INSERT INTO medicine
(name,frequency,time,timet,note,quantity,type,dose,refill) VALUES
("${nameController.text}","${int.parse(freqController.text)}","$timeString","$timeString2","${note
Controller.text}","${quantityController.text}","$Pill","${doseController.text}","${refillController.t
ext}")');
          await sqldb.insert(

```



```

        'INSERT INTO notes (medicine,note) VALUES
        ("${nameController.text}","${noteController.text}");

        print('=====');
        print(r);

        createScheduledNotification(
            int.parse(timeString.split(':')[0]),
            int.parse(timeString.split(':')[1]),
            nameController.text,
            r);

        createScheduledNotification(
            int.parse(timeString2.split(':')[0]),
            int.parse(timeString2.split(':')[1]),
            nameController.text,
            r + 100);

        Navigator.of(context).pushReplacementNamed('/home');
    } else if (nameController.text.isNotEmpty &&
        freqController.text.isNotEmpty &&
        quantityController.text.isNotEmpty &&
        noteController.text.isNotEmpty &&
        timeString != " " &&
        doseController.text.isNotEmpty &&
        refillController.text.isNotEmpty &&
        freqNumber != 2) {
        int r = await sqldb.insert(
            'INSERT INTO medicine
            (name,frequency,time,timet,note,quantity,type,dose,refill) VALUES
            ("${nameController.text}","${int.parse(freqController.text)}","$timeString","0","${noteController.t
            ext}","${quantityController.text}","$Pill","${doseController.text}","${refillController.text}");
            await sqldb.insert(
                'INSERT INTO notes (medicine,note) VALUES
                ("${nameController.text}","${noteController.text}");

            print('=====');
            print(r);

            createScheduledNotification(
                int.parse(timeString.split(':')[0]),
                int.parse(timeString.split(':')[1]),
                nameController.text,
                r);
            Navigator.of(context).pushReplacementNamed('/home');
        } else {
            final snackbar = SnackBar(
                behavior: SnackBarBehavior.floating,
                content: Text('Please Fill all Fields'),
                margin: EdgeInsets.all(10),
                duration: Duration(seconds: 3),

```



```

        backgroundColor: Color.fromRGBO(0, 0, 0, 0.7),
      );
      ScaffoldMessenger.of(context).showSnackBar(snackbar);    }},.),),), ), ), ), );}}

```

## Main:

```

main() {
  AwesomeNotifications().initialize('resource://drawable/pill', [
    NotificationChannel(
      channelKey: 'basic key',
      channelName: 'basic channel',
      channelDescription: 'notification for test',
      playSound: true,
      enableVibration: true,
      importance: NotificationImportance.High),
    NotificationChannel(
      channelKey: 'schedual_key',
      channelName: 'schedual notification',
      channelDescription: 'schedual',
      playSound: true,
      enableVibration: true,
      importance: NotificationImportance.High,
      locked: true,
      criticalAlerts: true),
  ]);

  SqlDb sql = SqlDb();
  actionEvent(sql);
  runApp(MyApp());
}

```

```

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      //home: HomePage(),
      initialRoute: '/',
      routes: {
        '/onboard': (context) => onboardPage(),
        '/': (context) => Splash(),
        '/addMedicin': (context) => addMedicinePage(),
        '/home': (context) => HomePage(),
        '/TextRecognition': (context) => textR_Page(),
        '/export': (context) => Export()
      },
    );
  }
}

```



## Homepage:

```
import 'package:get/get.dart';

import 'package:awesome_notifications/awesome_notifications.dart';
import 'package:flutter/material.dart';
import 'package:medically/card.dart';
import 'package:medically/notificationPage.dart';
import 'export.dart';
import 'sql.dart';
import 'textRecognitionPage.dart';
import 'widgets.dart';

class HomePage extends StatefulWidget {
  @override
  State<StatefulWidget> createState() => _HomaPageState();
}

Future<void> actionEvent(SqlDb sql) async {
  await AwesomeNotifications().setListeners(
    onActionReceivedMethod: (ReceivedAction notification) async {

    await sql.update(
      'UPDATE medicine SET quantity = quantity-dose WHERE id = "${notification.id}");
    List<Map> row = await sql
      .read('SELECT * FROM medicine WHERE id = "${notification.id}");
    if (row[0]['quantity'] <= row[0]['refill']) {
      AwesomeNotifications().createNotification(
        content: NotificationContent(
          id: notification.id! + 100,
          channelKey: "basic key",
          title: "Quantity Getting low",
          body:
            "Your Medicine ${row[0]['name']}' has Low Quantity Please Refill",
          notificationLayout: NotificationLayout.BigText,
          category: NotificationCategory.Message,
          fullScreenIntent: true));
    }
    //Navigator.of(context).popAndPushNamed('/home');
    Get.offNamed('/home');
    throw ("");
  });
}

bool Dark = true;
List<Map> medList = [];

class _HomaPageState extends State<HomePage> {
```



```

SqlDb sql = SqlDb();

List<Map> r = [];

Future<List<Map>> readData() async {
  List<Map> response = await sql.read('SELECT * FROM medicine');
  setState(() {
    medList = response;
  });
  return response;
}

@override
void initState() {
  AwesomeNotifications().requestPermissionToSendNotifications();

  readData();

  super.initState();
}

int barIndex = 0;
@override
Widget build(BuildContext context) {
  return Scaffold(
    bottomNavigationBar: Stack(children: [
      Container(
        decoration: Dark
          ? BoxDecoration(
              gradient: LinearGradient(
                begin: Alignment.topLeft,
                end: Alignment.bottomRight,
                colors: [
                  Color.fromARGB(255, 70, 68, 68),
                  Colors.black87
                ]
              )),
        : BoxDecoration(
              gradient: LinearGradient(
                begin: Alignment.topLeft,
                end: Alignment.bottomRight,
                colors: [
                  Colors.blue,
                  Color.fromARGB(255, 39, 132, 207)
                ]
              )),
        height: 60,
      ),
      BottomNavigationBar(
        elevation: 12,

```



```

type: BottomNavigationBarType.shifting,
selectedItemColor: Colors.white,
unselectedItemColor: Colors.grey,
items: [
  BottomNavigationBarItem(
    icon: Icon(Icons.home),
    label: 'Home',
    backgroundColor: Colors.transparent,
  ),
  BottomNavigationBarItem(
    icon: Icon(Icons.text_fields_sharp),
    label: 'Text Recognition',
    backgroundColor: Colors.transparent,
  ),
  BottomNavigationBarItem(
    icon: Icon(Icons.text_snippet_rounded),
    label: 'Export PDF',
    backgroundColor: Colors.transparent,
  ),
],
onTap: (index) {
  setState(() {
    barIndex = index;
  });
},
currentIndex: barIndex,
),
]),
appBar: AppBar(
  automaticallyImplyLeading: false,
  flexibleSpace: Container(
    decoration: Dark
      ? BoxDecoration(
        gradient: LinearGradient(colors: [
          Color.fromARGB(255, 70, 68, 68),
          Colors.black87
        ])
      : null),
  elevation: 6,
  title: Row(
    crossAxisAlignment: CrossAxisAlignment.center,
    children: [
      Text(
        'Medicaly',
        style: TextStyle(
          fontSize: 28,
          fontWeight: FontWeight.w900,
          color: Colors.white70),

```



```

    ),
    Image.asset(
      'assets/images/m.gif',
      height: 50,
      width: 50,
    )
  ],
),
actions: [
  IconButton(
    onPressed: () {
      setState(() {
        Dark = !Dark;
      });
    },
    icon: Dark == true
      ? Icon(Icons.sunny)
      : Icon(Icons.dark_mode_sharp))
],
),
body: barIndex == 0
  ? Container(
    decoration: Dark == true
      ? BoxDecoration(
        gradient: LinearGradient(colors: [
          Color.fromARGB(255, 70, 68, 68),
          Colors.black87
        ], begin: Alignment.bottomCenter, end: Alignment.topCenter))
      : null,
    child: RefreshIndicator(
      onRefresh: readData,
      child: ListView(
        physics: AlwaysScrollableScrollPhysics(),
        children: [
          ListView.builder(
            itemCount: medList.length,
            shrinkWrap: true,
            physics: NeverScrollableScrollPhysics(),
            itemBuilder: (context, i) {
              return Card1(
                medList[i]['name'].toString(),
                medList[i]['time'].toString(),
                medList[i]['timet'].toString(),
                medList[i]['frequency'],
                medList[i]['type'],
                medList[i]['quantity'],
                context,
                sql,

```





```

        medList[i]['id'],
        readData,
        medList[i]['note'],
        medList[i]['dose'].toString(),
        medList[i]['refill'].toString());
    },
  )
  ]),
),
)
: barIndex == 1
  ? textR_Page()
  : Export()

',
floatingActionButton: barIndex == 0
  ? SizedBox(
    width: 120,
    child: FloatingActionButton(
      shape: BeveledRectangleBorder(
        borderRadius: BorderRadius.circular(6)),
      backgroundColor: Dark ? Colors.grey[600] : null,
      child: Text('New Medicine'),
      onPressed: () {
        Navigator.of(context).pushReplacementNamed('/addMedicin');
      },
    ),
  )
  : null,
);
}
}

```

## APPENDIX B: ALGORITHM USED

### Text Recognition:

```
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:google_mlkit_text_recognition/google_mlkit_text_recognition.dart';
import 'dart:io';
import 'homePage.dart';
import 'package:google_mlkit_translation/google_mlkit_translation.dart';
import 'package:google_mlkit_language_id/google_mlkit_language_id.dart';

class textR_Page extends StatefulWidget {
  @override
  State<StatefulWidget> createState() => _textR_PageState();
}

class _textR_PageState extends State {
  final ImagePicker _picker = ImagePicker();
  String? myText;
  File? pickedImage;
  getImage() async {
    final XFile? image = await _picker.pickImage(source: ImageSource.camera);
    if (image == null) return;
    setState(() {
      pickedImage = File(image.path);
    });
  }

  Future<String?> recognizeText(InputImage image) async {
    try {
      TextRecognizer recognizer = TextRecognizer();
      RecognizedText recognizedText = await recognizer.processImage(image);
      recognizer.close();
      return recognizedText.text;
    } catch (error) {
      return null;
    }
  }
  ;
}

Future<String?> translateText(String text) async {
  final langident = LanguageIdentifier(confidenceThreshold: 0.5);
  final langCode = await langident.identifyLanguage(text);
  langident.close();
  final translator = OnDeviceTranslator(
    sourceLanguage: TranslateLanguage.values
      .firstWhere((element) => element.bcpCode == langCode),
```



```

        targetLanguage: TranslateLanguage.arabic);
    final response = await translator.translateText(text);
    translator.close();
    return response;
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    floatingActionButton: myText != null
      ? FloatingActionButton(
        backgroundColor: Dark ? Color.fromARGB(255, 77, 73, 73) : null,
        child: Icon(Icons.translate_rounded),
        onPressed: () async {
          final translatedT = await translateText(myText!);
          setState(() {
            myText = translatedT;
          });
          print(myText);
        })
      : null,

    body: Container(
      decoration: Dark == true
        ? BoxDecoration(
          gradient: LinearGradient(
            colors: [Color.fromARGB(255, 70, 68, 68), Colors.black87],
            begin: Alignment.bottomCenter,
            end: Alignment.topCenter))
        : null,
      child: ListView(
        children: [
          Column(
            children: [
              SizedBox(height: 20),
              Container(
                width: 400,
                decoration: BoxDecoration(
                  border: Border.all(width: 4, color: Colors.grey)),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                children: [
                  ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      shape: StadiumBorder(),
                      backgroundColor:
                        Dark ? Color.fromARGB(255, 77, 73, 73) : null,
                      elevation: 4),

```



```

        child: Text('Take Image'),
        onPressed: () async {
          getImage();
        },
      ),
      pickedImage != null
        ? Image.file(
            pickedImage!,
            width: 100,
            height: 100,
          )
        : Container(
            width: 100,
            height: 100,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  'Please Take Image',
                  style: TextStyle(
                    color:
                      Dark ? Colors.white : Colors.black),
                ),
              ],
            ),
          ),
    ],
  ),
  SizedBox(height: 60),
  Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          shape: StadiumBorder(),
          backgroundColor:
            Dark ? Color.fromARGB(255, 77, 73, 73) : null,
          elevation: 4),
        onPressed: () async {
          String? Text_After_Recog = await recognizeText(
            InputImage.fromFile(File(pickedImage!.path)));
          setState(() {
            myText = Text_After_Recog;
          });
        },
        child: Text('Generate text'))
    ],
  ),

```



```

    )
  ],
),
myText != null
  ? Container(
    width: 400,
    color: Colors.black12,
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Expanded(
          child: Text(
            myText!,
            style: TextStyle(
              fontSize: 30,
              color: Dark ? Colors.grey : Colors.black),
          ),
        )
      ],
    ),
  )
: Row()
],
),
),
);
}
}

```

### Export Medicine List:

```

import 'package:flutter/material.dart';
import 'pdf.dart';
import 'sql.dart';
import 'homePage.dart';

class Export extends StatefulWidget {
  @override
  State<StatefulWidget> createState() => _ExportState();
}

class _ExportState extends State<Export> {
  List<Map> response = [];
  readData() async {
    SqlDb sql = SqlDb();
    response = await sql.read('SELECT name FROM medicine');
  }
}

```



```

@override
void initState() {
  readData();
  super.initState();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    // appBar: AppBar(),
    body: Container(
      decoration: Dark
        ? BoxDecoration(
            gradient: LinearGradient(
              colors: [Color.fromARGB(255, 70, 68, 68), Colors.black87],
              begin: Alignment.bottomCenter,
              end: Alignment.topCenter))
        : null,
      child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'You Can Save A PDF File With Your Current Medicines ',
              style: TextStyle(
                fontWeight: FontWeight.bold,
                color: Dark ? Colors.white : Colors.black),
            ),
            SizedBox(height: 100),
            ElevatedButton(
              style: ElevatedButton.styleFrom(
                shape: StadiumBorder(),
                backgroundColor:
                  Dark ? Color.fromARGB(255, 77, 73, 73) : null,
                elevation: 4),
              onPressed: () async {
                final pdf1 = await pdf.generateText(response);
                print(response);
                pdf.openFile(pdf1);
              },
              child: Text('Save PDF')),
          ],
        ),
      ),
    );
}

```



## Notification:

```
import 'package:awesome_notifications/awesome_notifications.dart';

Future<void> createScheduledNotification(
  int hour, int minute, String title, int id) async {
  await AwesomeNotifications().createNotification(
    content: NotificationContent(
      id: id,
      channelKey: 'scheduling_key',
      title: title,
      body: 'its $title Time',
      notificationLayout: NotificationLayout.BigText,
      wakeUpScreen: true,
      category: NotificationCategory.Alarm,
      actionButtons: [NotificationActionButton(key: 'Done', label: 'Taken')],
      schedule: NotificationCalendar(
        hour: hour,
        minute: minute,
        second: 0,
        millisecond: 0,
        repeats: true,
        preciseAlarm: true,
        allowWhileIdle: true));
    }

import 'package:awesome_notifications/awesome_notifications.dart';
import 'package:flutter/material.dart';
import 'sql.dart';

class notificationPage extends StatelessWidget {
  SqlDb sql = SqlDb();
  int notificationMedicineid = 0;
  notificationPage(this.notificationMedicineid);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Container(
          child: ElevatedButton(
            onPressed: () async {
              await sql.update(
                'UPDATE medicine SET quantity = quantity-dose WHERE id =
                "${notificationMedicineid}");
              List<Map> row = await sql.read(
                'SELECT * FROM medicine WHERE id = "${notificationMedicineid}");
              if (row[0]['quantity'] <= row[0]['refill']) {
```



```
AwesomeNotifications().createNotification(  
  content: NotificationContent(  
    id: 1,  
    channelKey: "basic key",  
    title: "Quantity Getting low",  
    body:  
      "Your Medicine ${row[0]['name']} has Low Quantity Please Refill",  
    notificationLayout: NotificationLayout.BigText,  
    category: NotificationCategory.Message,  
    fullScreenIntent: true));  
  }  
  Navigator.pop(context);  
},  
child: Text('Taken'))),  
),  
);  
}  
}
```



## المستخلص

هدفنا هو تطوير تطبيق تذكير طبي شامل يلبي الاحتياجات المتنوعة لجميع المستخدمين. نتيح واجهتنا سهلة الاستخدام للمرضى إدخال المعلومات الطبية الحيوية وتتبعها بسهولة مثل جداول الأدوية والجرعات والترددات ومتطلبات إعادة التعبئة. يلغي تطبيقنا حاجة المرضى إلى تذكر الجرعات المميزة لكل دواء ، مما يبسط حياتهم ويقلل من التوتر. نتيح إمكانية الوصول إلى تطبيقنا للجميع ، بما في ذلك المرضى والمرضات وأفراد الأسرة ، التحكم في صحتهم من خلال توفير معلومات دقيقة ومحدثة متاحة في أي وقت وفي أي مكان.