ALERT MESSAGE USING DOCTOR'S PRESCRIPTION

A PROJECT REPORT

submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
KAKINADA

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

By

R.SAI VENKATA MEGHANA	**19HP1A0527**
M.APARNA SRI	**19HP1A0502**
SYED ZEHRA BANU	**20HP5A0506**

UNDER THE ESTEEMED GUIDANCE OF

Y.RAJESH

Assistant Professor, CSE Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ANDHRA LOYOLA INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Affiliated to JNTU KAKINADA)

Vijayawada 520008

APRIL 2023

ANDHRA LOYOLA INSTITUTE OF ENGINEERING AND TECHNOLOGY

VIJAYAWADA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "**ALERT MESSAGE USING DOCTOR'S PRESCRIPTION**" is submitted by **Ms. R Sai Venkata Meghana, Ms. M. Aparna Sri, and Ms. Syed Zehra Banu** to the JNTU KAKINADA in partial fulfillment for the award of Degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out by them under my supervision during the year 2022-2023.

(Y.Rajesh)	**(Dr. Ch Rajendra Babu)**
Assistant Professor	**Head of the Department**

Signature of the External Examiner

DECLARATION

We Ms.R.SAI VENKATA MEGHANA,Ms.M.APARNA SRI, Ms.SYED.ZEHRA BANU, hereby declare that the project report entitled "**ALERT MESSAGE USING DOCTOR'S PRESCRIPTION**" is an original work done in the Department of Computer Science and Engineering, Andhra Loyola Institute of Engineering and Technology, Vijayawada, during the academic year 2022-2023, in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering. We assure that this project is not submitted to any other University or College.

Roll No	Name of the student	Signature
19HP1A0527	R.S.V.MEGHANA	
19HP1A0502	M.APARNA SRI	
20HP5A0506	SD.ZEHRA BANU	

ACKNOWLEDGEMENT

Firstly, we would like to convey our heartful thanks to Almighty for the blessings on us to carry out this project work without any disruption.

We express our deep sense of gratitude to our Project Guide **Y.Rajesh , Assistant Professor, Department of Computer Science and Engineering (CSE), Andhra Loyola Institute of Engineering and Technology (ALIET), Vijayawada** for his able and inspiring guidance and valuable suggestions throughout our project work.

We are very thankful to our beloved Head of Department **Dr. Ch. Rajendra Babu, HOD Department of Computer Science and Engineering (CSE), Andhra Loyola Institute of Engineering and Technology (ALIET), Vijayawada,** and all other faculty members for providing their kind support throughout the course of Major-Project.

We are highly grateful to **Rev. Fr. Dr. Francis Xavier, Director** and also to **Dr.O. Mahesh, Principal, Andhra Loyola Institute of Engineering and Technology (ALIET), Vijayawada** without whose support and blessings would not have been possible for us to carry out the project work.

We are thankful to our project coordinator **Dr. Ch. Rajendra Babu, HOD Department of Computer Science and Engineering (CSE)** for his valuable guidance which helped us complete this project successfully.

Finally, we would like to convey our heartful thanks to our Technical Staff, for their guidance and support in every step of this project. We convey our sincere thanks to all the faculty and friends who directly or indirectly helped us for the successful completion of this project.

PROJECT ASSOCIATES

R. Sai Venkata Meghana	(19HP1A0527)
M.Aparna Sri	(19HP1A0502)
Syed Zehra Banu	(20HP5A0506)

ABSTRACT

Prescription is the medicine report given by a doctor to the patient when he/she is facing any health issue. Some people will forget to take the medicine on time. We want to develop an app "**Alert Message using doctor's prescription**" which is used to set the alarm from the prescription. Clicking the picture of the prescription, we will display the tablet details and set the alarm. If the doctor did not mention the patterns in that then by using the tablet details, we will set the alarm. The front end was implemented using android studio and the back end was designed using firebase framework. The android based application runs on mobile devices, such as smartphones, tablet computers. And patients can get the information of the disease by taking the symptoms from the user.

List Of Figures

TABLE OF CONTENTS

CHAPTER - 1

INTRODUCTION

1.1 ABOUT PROJECT:

The category of patients involve all human beings-teachers, students, businessmen, housewives, children and also all of us have a busy hectic schedule. Today’s life is full of responsibilities and stress. So people are prone to diseases of different types and it is our duty to make ourselves stay fit and healthy. If the patient stays at home then he or she might get someone to look after him/her but when one is not at home, is out of the city or state away from home then it is hard for the family members to call them and remind them their dosage timings every time.

When it comes to patient care, health is a critical factor to consider. Patients who are in good health generally have a better prognosis and may respond more positively to treatment. On the other hand, patients who are in poor health may have a more challenging recovery process and may be at higher risk of complications.

Health and patients are also linked in terms of prevention. Maintaining good health can help to prevent the development of many illnesses and health conditions, which can ultimately reduce the need for medical treatment. For example, individuals who eat a healthy diet, exercise regularly, and avoid smoking and excessive alcohol consumption may be less likely to develop chronic conditions such as heart disease, diabetes, and certain types of cancer.

In terms of patient care, prevention is also important. Health care providers may work with patients to develop strategies for maintaining good health and preventing the development of future health problems. This may include lifestyle changes, such as dietary modifications and exercise, as well as regular check-ups and screenings to identify potential health concerns early on.

To develop an android system that will useful for:

- Users can directly get notification or alert ringing to remind them.
- Information about the problem facing them by gathering the symptoms.

1.2 PURPOSE OF THE PROJECT:

The main objective is to remind patients about taking their prescribed medicines or pills within the stipulated time as prescribed by a doctor and predicting the diseases by taking information from the user.

MEDMINDER uses optical character recognition (OCR) technology to extract relevant tablet details from a picture of the prescription and set reminders based on the information provided by the doctor.

The app allows users to receive SMS or email reminders based on their preferences and also allows users to find hospitals and doctors based on their location and update their profiles with relevant medical information.

1.3 SCOPE OF THE PROJECT:

The application would help patients to maximize the full benefit of the medicine and abstain them from the risk that result as not taking the medicine or pill within the time. It will be of great help for patients suffering from a range of problems such as forgetfulness, busy schedules, old age.

CHAPTER-2

LITERATURE SURVEY

PAPER : Development of an Android Based Medication Reminder and Adherence System

AUTHORS : Adeyemi, T.O.

ANALYSIS : The proposed system is useful to patients to set a customized alarm tone in their local language or select from a list of default tones and allows specialists to automatically see the list of patients connected to them and their chat messages .And assist patients with chronic illness like Cancer, Diabetes, Asthma and HIV/AIDS, to get notifications from medical personnel about the availability of drugs and also served as a reminder system,

PAPER : Medication Reminder And HealthCare Application

AUTHORS : Deepti Ameta, Kalpana Mudaliar and Palak Patel

ANALYSIS : The proposed system is based on Android Operating system which will remind the users to take medicines on time through notification and automatic alarm ringing system. The users will get the schedule of medicine in-take time with medicine description, starting and ending date of medicine, notification through message or email, automatic alarm ringing system and navigation system.

PAPER : Medical Handwritten Prescription Recognition Using CRNN

AUTHORS : Rayan Haidar , Roger Achkar

ANALYSIS : The proposed system established a Convolutional Recurrent Neural Network (CRNN) technology using Python that can interpret handwritten English prescriptions and translate them into digital text. For this, datasets with 66 different classes, including alphanumeric characters, punctuation, and spaces, were used. Since prescriptions generally contain two or three words, the training was carried out using short texts. Normal handwriting and prescriptions from doctors were used to train the model. The system got a 98% accuracy rate after taking training time and data input into account.

PAPER : Pill Dispenser with alarm Via Smartphone notification

AUTHORS : Nurmiza Binti Othman

ANALYSIS : This proposed system is a pill dispenser that can ensure that the medication is taken safely and on time, especially among the elderly and the alarm function was added to the system as a popup notification . Combination of infrared sensor and Arduino microcontroller control the medication dose. Alarm system is implemented via popup notification on the user's smartphone.

CHAPTER-3

SYSTEM ANALYSIS

3.1 FEASABILITY STUDY

The feasibility of the project is analyzed in this phase and is put forth with a very general plan for the project and some cost estimates. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are:

- Economical Feasibility
- Operational Feasibility
- Technical Feasibility

3.1.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.1.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 EXISTING SYSTEM :

Patients has to type their medicine and schedule the reminder to take medicines. These cause the wrong typing medicine or set wrong time .By that it may causes health in danger.

Drawbacks of the Existing System

- It is a manual set reminder.
- He won’t get sms and email reminder at that time.
- Doesn’t give description about medicine details.

3.3 PROPOSED SYSTEM :

People most of them facing difficulty in taking on time and unable to understand medicines. So we want to propose an android application which can be used for setting the alarm from prescriptions itself and give the notifications by various ways such as Sound, Messages via through email or normal message.

By combining the android with Artificial Neural Network we going to develop an app which gives alert for the patients with medicine prescription and we can also include another module like predicting the disease by using symptoms from the patient.

CHAPTER-4

SYSTEM REQUIREMENTS

SOFTWARE SPECIFICATIONS:

• Operating system	:	Windows 10/11,Ubuntu
• Coding Language	:	Java
• Data Base	:	Firebase
• Model Design	:	Rational rose
• IDE	:	Android Studio

MINIMUM HARDWARE SPECIFICATIONS:

• Processor	:	Intel i3,i5,i7
• Memory	:	8 GB
• Hard Disk	:	250 G

CHAPTER-5

SYSTEM DESIGN

5.1 INTRODUCTION:

Design is the first step in the development phase for any techniques and principles for
the purpose of defining a device, a process or system in sufficient detail to permit its
physical realization.

Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer's requirements into finished software or a system.

Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

5.2 UML DIAGRAMS

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. It is based on diagrammatic representations of software components. UML has been used as a general-purpose modeling language in the field of software engineering.

Types of UML Diagrams

There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after.The two most broad categories that encompass all other types are Behavioral UML diagram and Structural UML diagram.

5.2.1 USECASE DIAGRAM

The use case diagrams of the UML are used for the requirements analysis in the design phase of the system. Here the entire system is depicted in terms of use cases and actors. The actors interact with the system either directly or indirectly. Use cases
define the way how the system responds to the external factors i.e. with actors.

5.2.1.1 Purpose of Use Case Diagrams

The purpose of use case diagram is to capture the dynamic aspect of a system.
However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and Statechart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.
Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.
When the initial task is complete, use case diagrams are modeled to present the outside view.
In brief, the purposes of use case diagrams can be said to be as follows –

● Used to gather the requirements of a system.

● Used to get an outside view of a system.

● Identify the external and internal factors influencing the system.

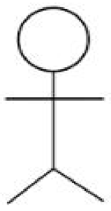
● Show the interaction among the requirements is actors

5.1.1.2 IDENTIFYING ACTORS

Identifying the actors is as important as identifying classes, structures, associations, attributes and behavior. The term actor represents the role a user plays with respect to the system. When dealing with actors, it is important to think about roles rather than people or job titles.

Actor:

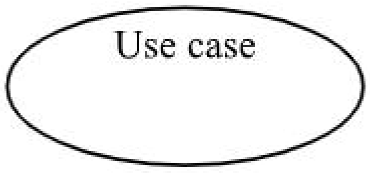
Actor represents the role a user plays with respect to the system. An Actor interacts with, but has no control over the use cases.
Graphical representation:



Actor

Use cases:

A use case describes a sequence of actions that provide something of measurable
value to an actor and is drawn as a horizontal ellipse.



Use case

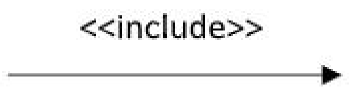
A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not. Four relationships among use cases are used often in practice.

Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends". A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.

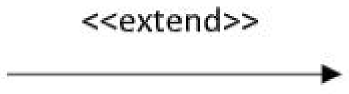
Include

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label &quot;«include»". There are no parameters or return values.



Extend

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label “«extend»”. Modelers use the «extend» relationship to indicate use cases that are “optional” to the base use case.



Generalization

In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation).

Associations

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

Identified Use Cases

Now it’s time to identify the use cases. A good way to do this is to identify what the actors needs from the system. So all of these can be considered as use cases. Top level use cases should always provide a complete functions required by an actor. You can extend or include use cases depending on the complexity of the system.Once you identify the actors and the top level use case you have basic idea of the system. Now you can fine tune it and add extra layers of detail to it.

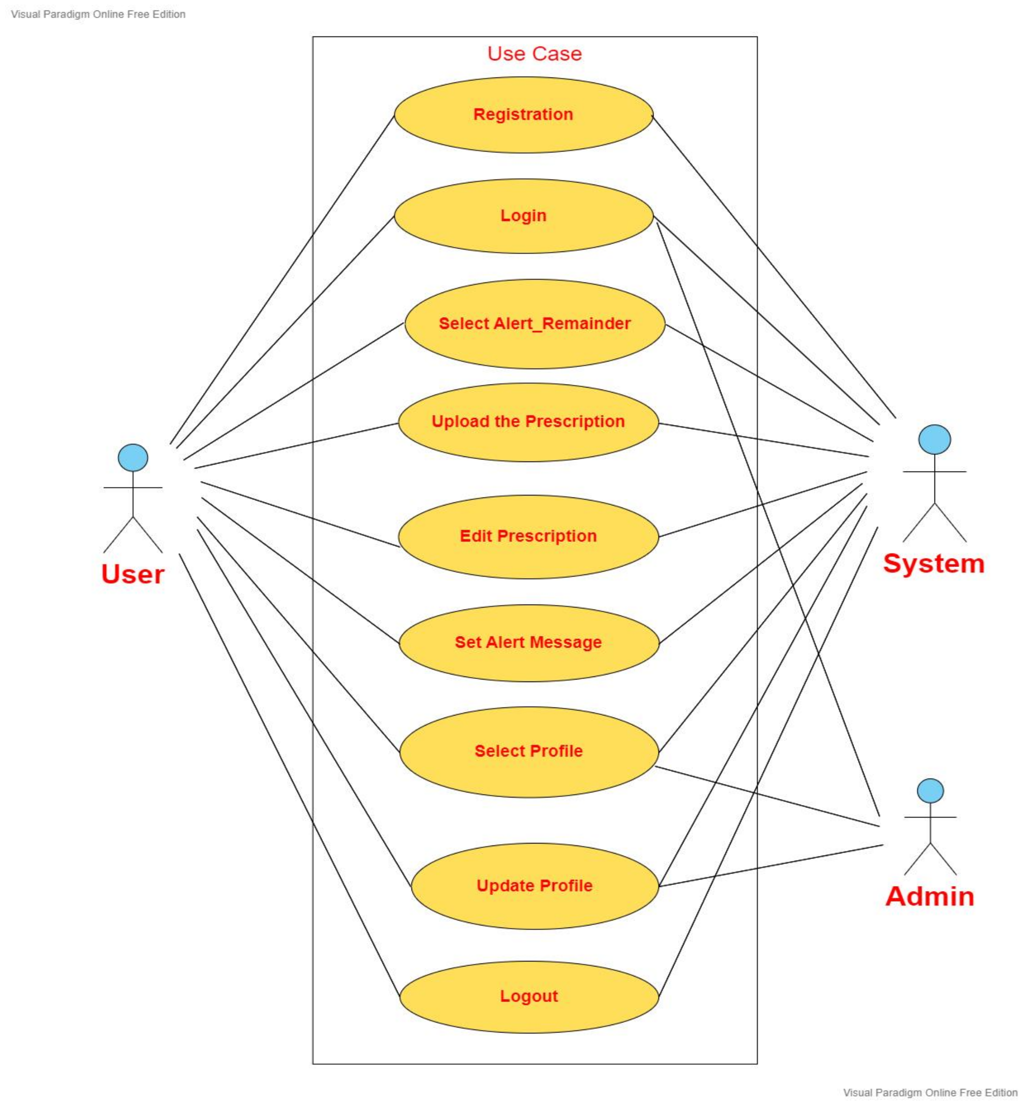


Figure 5.2.1.1 Use Case Diagram

5.2.2 INTERACTION DIAGRAMS

From the term Interaction, it is clear that the diagram is used to describe some type of interactions among the different elements in the model. This interaction is a part of dynamic behavior of the system.

Interaction diagrams are models that describe how a group of objects collaborate in some behavior - typically a single use-case. The diagrams show a number of example objects and the messages that are passed between these objects within the use-case.

Interaction diagrams come in two forms, both present in the UML:

- Sequence Diagrams
- Collaboration Diagrams

Purpose of Interaction Diagrams

The purpose of interaction diagrams is to visualize the interactive behavior of the system. Visualizing the interaction is a difficult task. Hence, the solution is to use

different types of models to capture the different aspects of the interaction.
Sequence and collaboration diagrams are used to capture the dynamic nature but from a different angle.
The purpose of interaction diagram is −

- To capture the dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe the structural organization of the objects.
- To describe the interaction among objects.

5.2.2.1 SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple run time scenarios in a graphical manner.

If the lifeline is that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances. Messages, written with horizontal arrows with the message name written above them, display interaction.
Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine.
If a caller sends an asynchronous message, it can continue processing and doesn’t have to wait for a response. Asynchronous calls are present in multi threaded applications and in message-oriented middleware. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML).

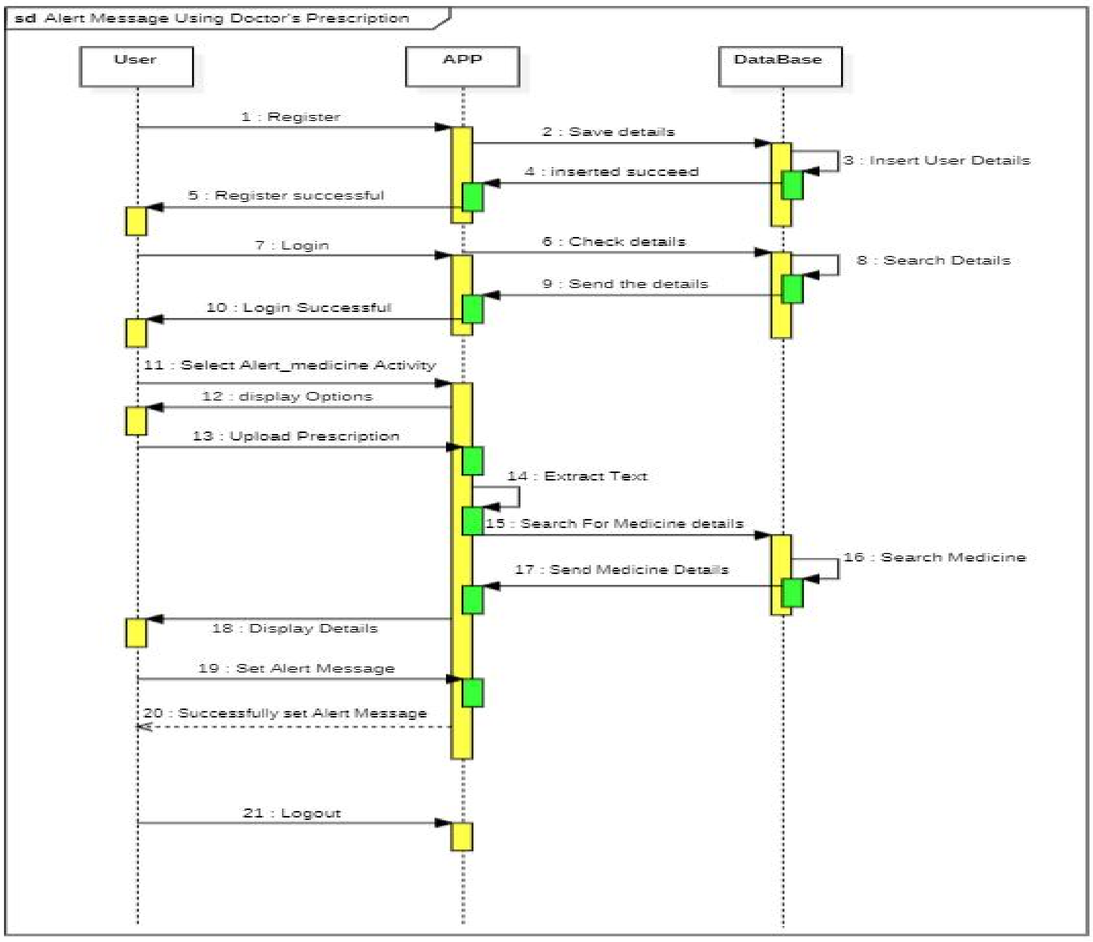


Figure 5.2.2.1 Sequence Diagram

5.2.2.2 COLLOBORATION DIAGRAM

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time . Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult to read. Several vendors offer software for creating and editing collaboration diagrams.

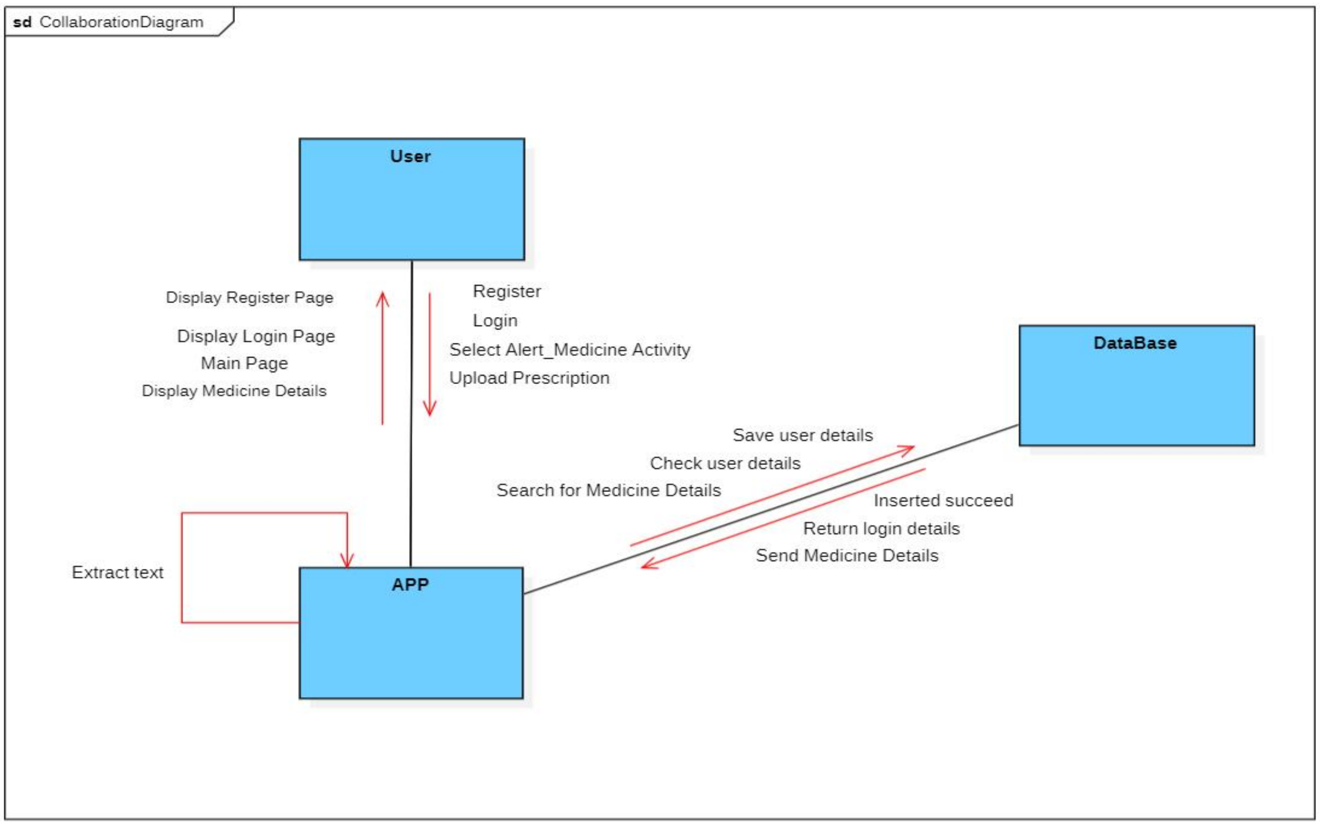


Figure 5.2.2.2 Colloboration Diagram

5.2.3 CLASS DIAGRAM

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design.

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

A class with three sections:

In the diagram, classes are represented with boxes which contain three parts:

● The upper part holds the name of the class

● The middle part contains the attributes of the class

● The bottom part gives the methods or operations the class can take or undertake.

Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as −

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

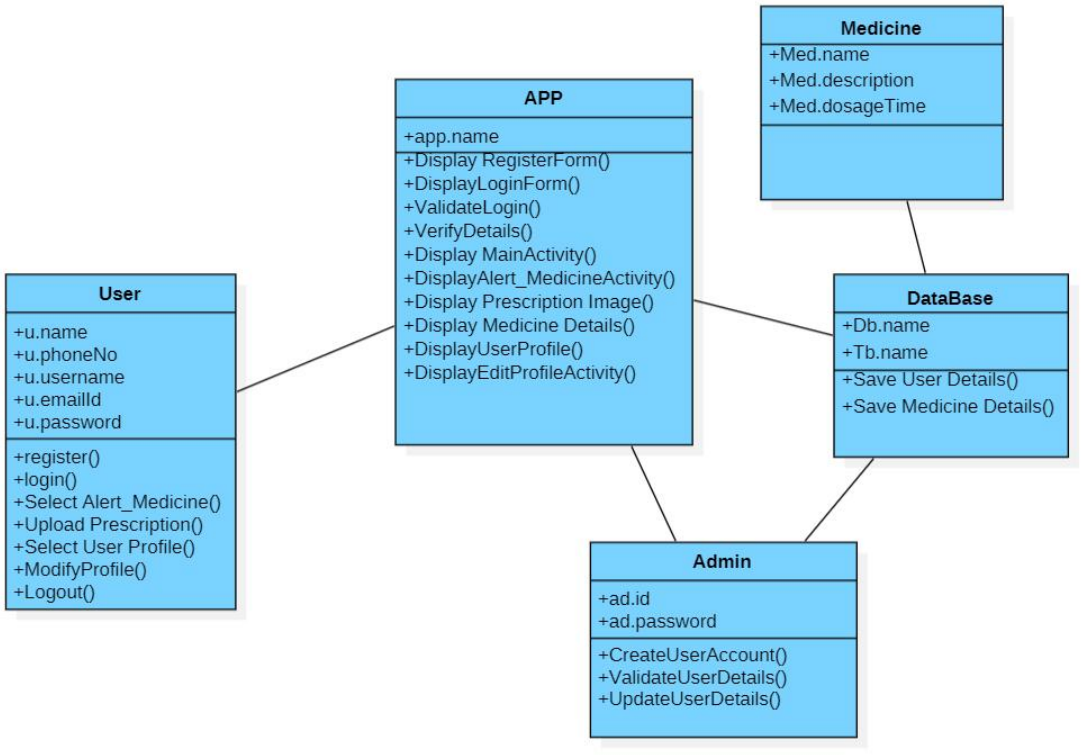


Figure 5.2.3 Class Diagram

5.2.4 ACTIVITY DIAGRAM

An activity diagram is a variation or special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations. An activity diagram can be used to model an entire business process. The purpose of an activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. However, activity diagram can also be used to represent a class‘s method implementation.

An activity model is similar to a state chart diagram, where a token represents an operation. An activity is shown as a round box, containing the name of the operation. When an operation symbol appears within an activity diagram or other state diagram, it indicates the execution of the operation. Executing a particular step within the diagram represents a state within the execution of the overall method.

The same operation name may appear more than once in a state diagram, indicating the invocation of the same operation in different phases. An outgoing solid arrow attached to an activity symbol indicates a transition triggered by the completion of the activity. The name of the implicit event need not be written, but the conditions that depend on the result of the activity or other values may be included.

An activity diagram is used mostly to show the internal state of an but external events may appear in them. An external event appears when the object is in a ―wait state‖, a state during which there is no internal activity by the object and the object is waiting for some external event to occur as the result of an activity by another object. The two states are wait state and activity state. More than the one possible event might take the object out of the wait state; the first one that occurs triggers the transition. A wait state is the ―normal state.

The purpose of an activity diagram can be described as −

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

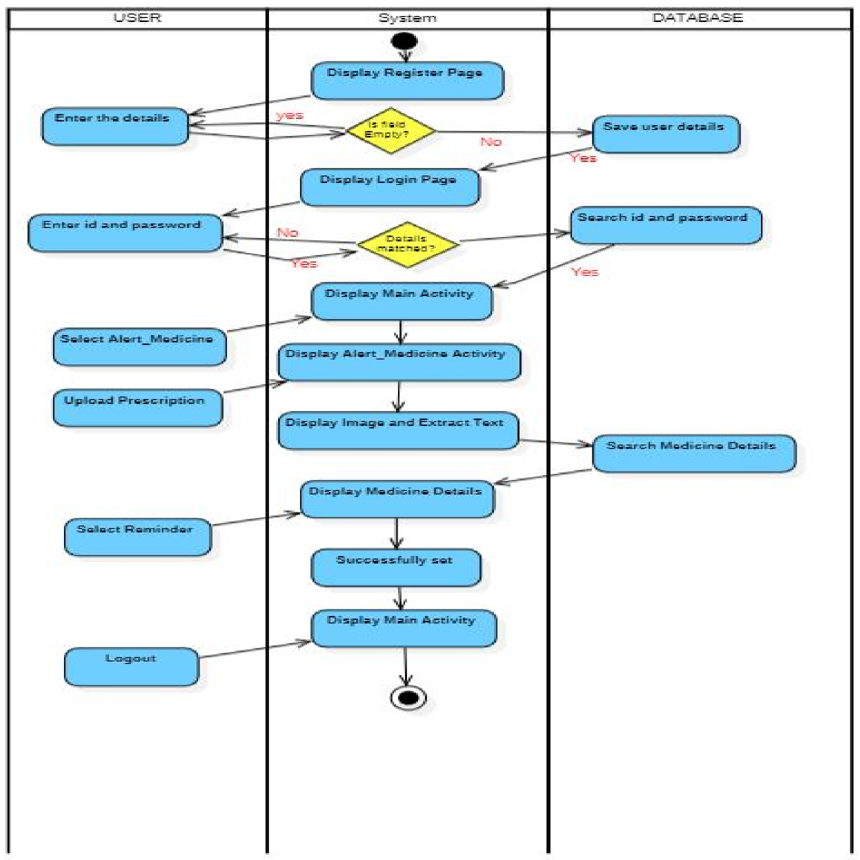


Figure 5.2.4 Activity Diagram

5.2.5 STATE CHART DIAGRAM

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. Activity diagram is a special kind of a State chart diagram. As State chart diagram defines the states, it is used to model the lifetime of an object.

Purpose of State chart Diagrams

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination.

State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using State chart diagrams –

● To model the dynamic aspect of a system.

● To model the life time of a reactive system.

● To describe different states of an object during its life time.

● Define a state machine to model the states of an object.

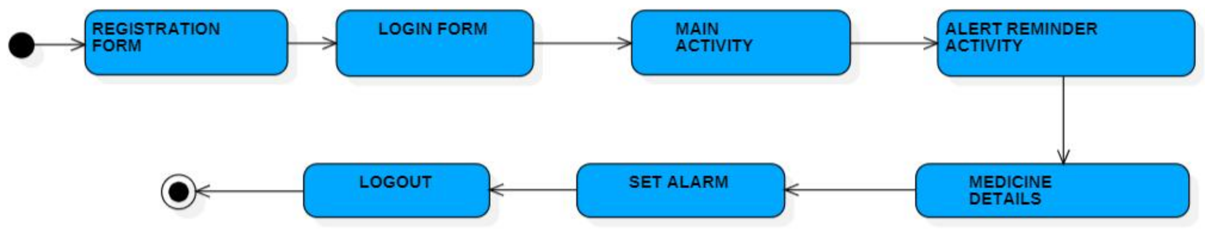


Figure 5.2.5 State Chart Diagram

5.2.6 COMPONENT DIAGRAM

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model physical aspects of a system. Now the question is what are these physical aspects? Physical aspects are the elements like executables, libraries,files, documents etc which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose of the component diagram

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc.

A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

So the purpose of the component diagram can be summarized as:

● Visualize the components of a system.

● Construct executables by using forward and reverse engineering.

● Describe the organization and relationships of the components.

Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executable, libraries etc. So the purpose of this diagram is different, Component diagrams are used during the implementation phase of an application.

This diagram is very important because without it the application cannot be implemented efficiently. A well prepared component diagram is also important for other aspects like application performance, maintenance etc.

So before drawing a component diagram the following artifacts are to be identified clearly:

● Files used in the system.

● Libraries and other artifacts relevant to the application.

● Relationships among the artifacts.

Now after identifying the artifacts the following points needs to be followed:

● Use a meaningful name to identify the component for which the diagram is to be drawn.

● Prepare a mental layout before producing using tools.

● Use notes for clarifying important points.

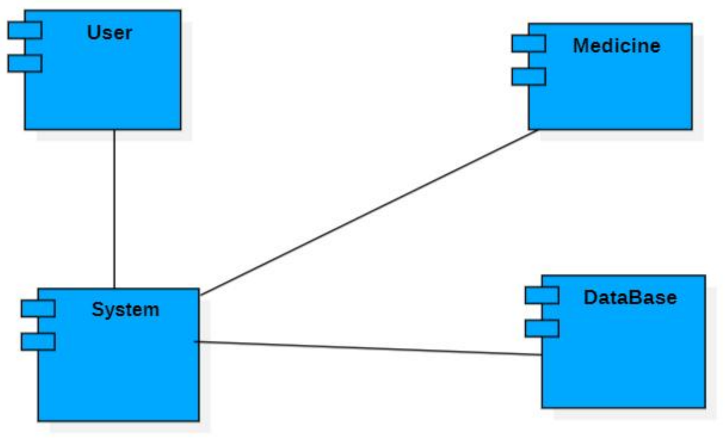


Figure 5.2.6 Component Diagram

5.2.7 DEPLOYMENT DIAGRAM

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

When to Use Deployment Diagram

● What existing systems will the newly added system need to interact or integrate with?

● How robust does system need to be (e.g., redundant hardware in case of a system failure)?

● What and who will connect to or interact with system, and how will they do it

● What middleware, including the operating system and communications approaches and protocols, will system use?

● What hardware and software will users directly interact with (PCs, network computers, browsers, etc.)?

● How will you monitor the system once deployed?

● How secure does the system needs to be (needs a firewall, physically secure hardware, etc.)?

Purpose of Deployment Diagrams

● They show the structure of the run-time system

● They capture the hardware that will be used to implement the system and the links between different items of hardware.

● They model physical hardware elements and the communication paths between them

● They can be used to plan the architecture of a system.

● They are also useful for Document the deployment of software components or nodes

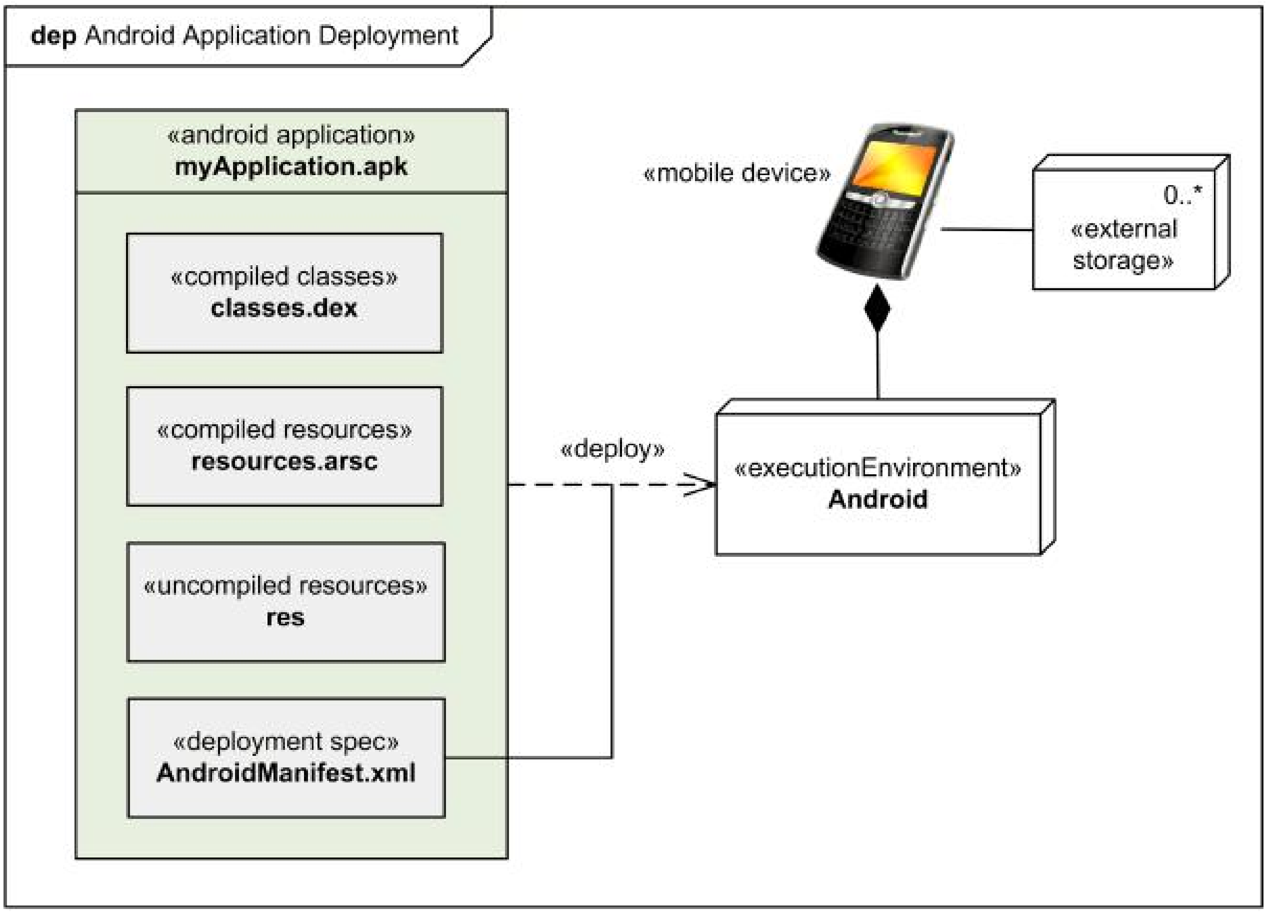


Figure 5.2.7 Deployment Diagram

CHAPTER-6

TECHNOLOGY DESCRIPTION

6.1 Android:

Android is a mobile operating system based on Linux and open source for smartphones and tablets.The Open Handset Alliance lead by Google and other businesses created Android. Android.Android provides a single application development method for mobile devices, allowing developers to only build and execute on Android-powered devices.Android's apps should work.

The first beta version of the Android Software Development Kit (SDK) was provided by Google in 2007 and was launched in September 2008 as the first commercial version, Android 1.0. Google unveiled the latest Android version, 4.1Jelly Bean, on June 27, 2012, during the Google I/O conference. Jelly Bean has been updated incrementally with the main objective, both functionality, and performance, to improve the user experience. Free and open-source software licenses are available for the source code for Android.

Under the GNU General Public License version, 2 Google releases the majority of its code in Apache 2.0 version, with the remainder changing the Linux kernel. Use the Android software development kit to generate Android applications in the Java language. Once the Android apps are built, they may simply be packaged and distributed through Google Play, SlideME, the Opera Mobile Store, Mobango, F-droid, and the Amazon Appstore.

In more than 190 countries across the world, Android power hundreds of millions of mobile devices. It is the largest mobile platform installed base and is expanding quickly. More than 1 million new Android smartphones are enabled throughout the globe every day.

Google Firebase:

Google Firebase is a Google-backed application development software that enables developers to develop IOS, Android, and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

Firebase offers several services, including:

Analytics – Google Analytics for Firebase offers free, unlimited reporting on as many as 500 separate events. Analytics presents data about user behavior in IOS and Android apps, enabling better decision-making about improving performance and app marketing.

Authentication – Firebase Authentication makes it easy for developers to build secure authentication systems and the feature offers a complete identity solution, supporting email and password accounts, phone authentication, as well as Google, Facebook, GitHub, Twitter login, and more.

Cloud messaging – Firebase Cloud Messaging(FCM) is a cross-platform messaging tool that lets companies reliably receive and deliver the message on IOS, Android, and web at no cost.

Realtime database – The Firebase Realtime Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real-time. The data is synced across all clients in real-time and is still available when an app goes offline.

Crashlytics – Firebase Crashlytics is a real-time crash reporter that helps developers track, prioritize, and fix stability issues that reduce the quality of their apps. With crashlytics, developers spend less time organizing and troubleshooting crashes and more time building features for their apps.

Performance – Firebase Performance Monitoring service gives developers insight into the performance characteristics of their IOS and Android apps to help them determine where and when the performance of their apps can be improved.

Test lab –Firebase Lab is a cloud-based app-testing infrastructure. With one operation, developers can test their IOS and Android apps across a variety of devices and device configurations. They can see the results, including videos, screenshot and logs, in the Firebase console.

6.2 Java:

Java is a programming language that is commonly used in the development of Android applications. Android Studio, an integrated development environment (IDE) for Android, provides developers with the tools necessary to create, test, and debug Android applications written in Java.
One of the key advantages of using Java in Android Studio is its ease of use. Java is a widely used and well-documented language, making it easy for developers to find answers to their questions and troubleshoot problems. Additionally, the Java programming language provides a rich set of libraries and frameworks that can help developers speed up the development process and reduce the amount of code they need to write.

Another advantage of using Java in Android Studio is the ability to write code that is compatible with a wide range of devices. Java's platform independence allows developers to write code that can run on multiple devices without needing to be recompiled or modified. This makes it easier for developers to create applications that can run on a wide range of Android devices, from low-end smart phone to high-end tablets.
Android Studio also provides a range of tools and features that make it easy for developers to create high-quality applications. For example, the IDE includes a layout editor that allows developers to create user interfaces for their applications visually, without needing to write code. Additionally, Android Studio includes a range of tools for debugging and profiling code, making it easier for developers to identify and fix bugs in their applications.
In summary, Java is a powerful and widely used programming language that is well-suited for developing Android applications in Android Studio. Its ease of use, compatibility with multiple devices, and rich set of libraries and frameworks make it a popular choice among Android developers. And with the wide range of tools and features available in Android Studio, developers have everything they need to create high-quality, robust Android applications using Java.

6.3 SQLite Database:

SQLite is a popular open-source relational database management system. It is a software library that provides a SQL database engine, which allows users to create, store, and manipulate relational databases. SQLite is widely used in many different applications and systems, including mobile devices, desktop applications, and web applications.

SQLite databases are stored as files on disk, making them easy to manage and move between systems. This makes SQLite a popular choice for applications that need to store and manage data locally. SQLite databases can also be used in client-server applications, where the database is hosted on a server and accessed by multiple clients.

One of the key advantages of SQLite is its lightweight design. The SQLite engine is small and efficient, requiring very little system resources to run. This makes SQLite a good choice for applications with limited system resources, such as mobile devices or embedded systems.

SQLite databases are also highly scalable. Users can create multiple tables within a single database file, allowing them to store and manage large amounts of data. SQLite databases support all standard SQL operations, including querying, updating, and deleting data.

Another advantage of SQLite is its cross-platform support. SQLite is available on all major operating systems, including Windows, macOS, Linux, and mobile platforms such as iOS and Android. This makes it easy to create applications that work across different platforms and devices.

Overall, SQLite is a powerful and versatile relational database management system that is well-suited for a wide range of applications. Its lightweight design, scalability, and cross-platform support make it an ideal choice for applications that require a reliable and efficient database management system.
SQLite is a popular open-source relational database management system that offers many advantages over other database systems. Some of the key advantages of SQLite include:

Lightweight design: SQLite is a small, lightweight database engine that requires very little system resources to run. This makes it an ideal choice for applications with limited resources, such as mobile devices and embedded systems.

Cross-platform support: SQLite is available on all major operating systems, including Windows, macOS, Linux, and mobile platforms such as iOS and Android. This makes it easy to create applications that work across different platforms and devices.

Easy to use: SQLite is simple to use and easy to set up. Users can create a new database file with just a few lines of code, and the database can be accessed using standard SQL commands.

Transactional support: SQLite supports transactions, allowing users to ensure the integrity of their data and avoid data corruption in the event of a failure or error.

Self-contained: SQLite databases are self-contained, meaning that all data is stored in a single file. This makes it easy to move and manage the database between systems.

High performance: Despite its lightweight design, SQLite is a high-performance database system that can handle large datasets and complex queries.

Open-source: SQLite is open-source software, which means that it is free to use and can be modified and distributed by anyone. This makes it an ideal choice for open-source projects and applications.

Overall, the advantages of SQLite make it a popular choice for many different applications and systems. Its lightweight design, cross-platform support, and ease of use make it an ideal choice for mobile and embedded systems, while its high performance and transactional support make it well-suited for enterprise applications and data-intensive applications.

CHAPTER-7

METHODOLOGY

This work was designed using the integration of a real-time mobile-based communication among multiple users on a single platform. It is strictly developed to run on android powered smartphones. The front end is designed using android studio and the back end is designed using firebase framework. The software engineering methodology used for this work is the agile approach. Agile methodology is a practice that promotes continuous iteration of development and testing throughout the software development life cycle of the project. The Agile Process differs from the waterfall model in ways like: instead of having big projects, it attempts to break projects into small increments in a cyclic manner and it allows future updates to software applications.

The above diagram represent the process how an Optical Character Recognition(OCR) will extract the information from the image and how it divides the segments in the sentence. The system can be developed by Java ,OCR as front end, and firebase,SQLite Database as backend. These work can be designed by using integration of a real time mobile based communication among multiple users on a single platform. It is strictly developed to rum on android powered smartphones.

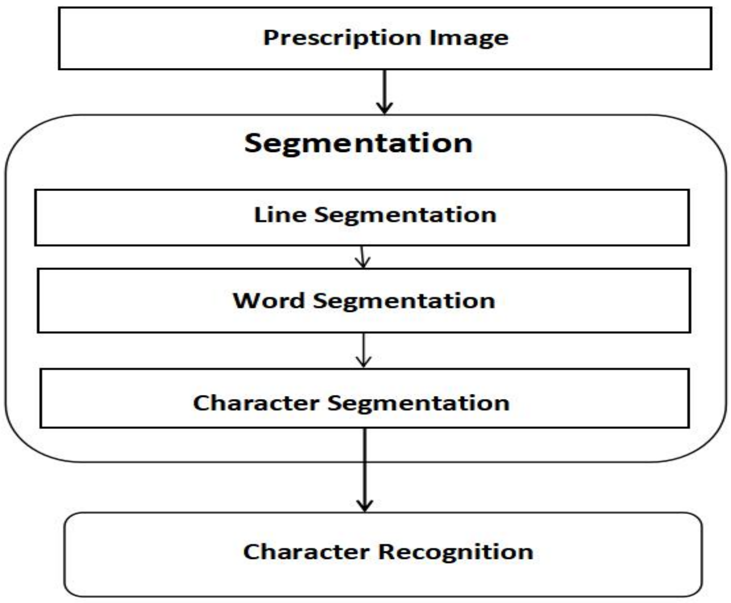


Figure 7.1 OCR Process for extracting Medicine

Many of these systems require special hardware devices to remind the patients about the medicine in-take timings. Purchasing new hardware devices becomes costly and more time and money consuming. So in the given work an attempt has been made to implement a system which is economical, easily accessible and improves medication reminder.

First we upload the prescription image and then model segment the written text into line and then words and character then it will set the alarm by find out the pattern inside of the text and it automatically set the alert message and sent to a mobile number or email .

CHAPTER-8

CODE

8.1 Java Class Code:

8.1.1 Android Manifest :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
    <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="com.example.permission.MAPS_RECEIVE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
        android:maxSdkVersion="32" />
    <uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES" />
  <uses-permission
android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        android:maxSdkVersion="32"
        tools:ignore="ScopedStorage" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.BIND_EXTERNAL_SERVICE" />
    <uses-permission
        android:name="android.permission.CALL_PHONE"
    tools:ignore="PermissionImpliesUnsupportedChromeOsHardware" />
  <uses-feature android:name="android.hardware.camera" />
  <uses-feature android:name="android.hardware.camera.autofocus"
    />
  <uses-feature
    android:name="android.hardware.telephony"
    android:required="false" />
  <application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="MedMinder"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.MyApplication"
    android:usesCleartextTraffic="true"
    tools:targetApi="31">
    <activity
      android:name=".about_us"
      android:exported="false" />
    <activity
      android:name=".DetailsMedicineActivity"
      android:exported="true"
      android:parentActivityName=".AlarmView" >
      <intent-filter>
        <action android:name="details" />
        <category android:name="android.intent.category.DEFAULT" />
      </intent-filter>
    </activity>
    <activity
      android:name=".AlarmView"
      android:exported="false"
      android:parentActivityName=".MainActivity" />
    <activity android:name=".NotificationActivity" />
    <activity android:name=".NotificationDetailsActivity" />
    <activity
      android:name=".SetAlarm"
      android:exported="true"
    android:parentActivityName=".ResultActivity" />
<activity
    android:name=".nearest_doctor"
    android:exported="false"
    android:parentActivityName=".MainActivity" />
<activity
    android:name=".nearest_hospital"
    android:exported="false"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.app.default_searchable"
        android:value=".activities.ListActivity" />
</activity>
<activity
    android:name=".ListActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.searchable"
        android:resource="@xml/searchable" />
</activity>
<receiver android:name=".recicver.AlarmReceiver" />
<receiver
    android:name=".recicver.SampleBootReceiver"
    android:enabled="true"
    android:exported="true"
    >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>

<service android:name=".AlarmRing"
    android:exported="true">
</service>
<service android:name=".SMSSenderService"/>
<activity
    android:name=".DetailActivity"
    android:exported="false" />
<meta-data
```

```
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyDMj8AgPlYraXNe5nNqjZAx4zE0onLNr5M" />
    <activity
        android:name=".ResultActivity"
        android:exported="false"
        android:parentActivityName=".alert_medicine" />
    <activity
        android:name=".SplashScreen"
        android:exported="true"
        android:theme="@style/SplashScreen">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".contact_us"
        android:exported="false"
        android:parentActivityName=".MainActivity">
        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
    <activity
        android:name=".call"
        android:exported="false"
        android:parentActivityName=".MainActivity">
        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
    <activity
        android:name=".symptom_checker"
        android:exported="false"
        android:parentActivityName=".MainActivity">
        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
```

```
</activity>
<activity
    android:name=".alert_medicine"
    android:exported="false"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
<activity
    android:name=".EditProfileActivity"
    android:exported="false"
    android:parentActivityName=".profile">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
<activity
    android:name=".profile"
    android:exported="false"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
<activity
    android:name=".LoginActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
<activity
    android:name=".RegisterActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
```

```
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
            <intent-filter>
                <action android:name="android.intent.action.VIEW"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <data android:scheme="file" android:host="*"/>
                <data android:scheme="content" android:host="*"/>
                <data android:mimeType="application/vnd.android.package-archive"/>
            </intent-filter>
        </activity>
        <meta-data
            android:name="com.google.android.gms.vision.DEPENDENCIES"
            android:value="ocr" />
        <activity android:name="com.theartofdev.edmodo.cropper.CropImageActivity"
            android:theme="@style/Base.Theme.AppCompat"
            tools:ignore="MissingClass" /> <!-- optional (needed if default theme has no action bar) -->
        <provider
            android:name="androidx.core.content.FileProvider"
            android:authorities="${applicationId}.fileprovider"
            android:grantUriPermissions="true"
            android:exported="false">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/provider_paths"/>
        </provider>
    </application>
</manifest>
```

8.1.2 LoginActivity.class :

```
package com.example.myapplication;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;
public class LoginActivity extends AppCompatActivity {
    EditText loginUsername, loginPassword;
    Button loginButton;
    TextView signupRedirectText;
    DatabaseReference reference;
    Query checkUserDatabase;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        loginUsername = findViewById(R.id.login_username);
        loginPassword = findViewById(R.id.login_password);
        loginButton = findViewById(R.id.login_button);
        signupRedirectText = findViewById(R.id.signupRedirectText);
        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(!validateUsername() | !validatePassword()) {
                } else {
                    checkUser();}
            }
        });
        signupRedirectText.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(LoginActivity.this,RegisterActivity.class)); }
        });
    }
    public Boolean validateUsername() {
        String val = loginUsername.getText().toString();
        if (val.isEmpty()) {
            loginUsername.setError("Username cannot be empty");
            return false;
        } else {
            loginUsername.setError(null);
            return true;}
    }
    public Boolean validatePassword(){
        String val = loginPassword.getText().toString();
        if (val.isEmpty()) {
            loginPassword.setError("Password cannot be empty");
            return false;
        } else {
            loginPassword.setError(null);
            return true; }
    }
    public void checkUser(){
        String userUsername = loginUsername.getText().toString().trim();
        String userPassword = loginPassword.getText().toString().trim();
        reference = FirebaseDatabase.getInstance().getReference("users");
        checkUserDatabase = reference.orderByChild("username").equalTo(userUsername);
        checkUserDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()){
                    loginUsername.setError(null);
                    String                    passwordFromDB                    =
snapshot.child(userUsername).child("password").getValue(String.class);
                    if (passwordFromDB.equals(userPassword)) {
                        loginUsername.setError(null);
                        String nameFromDB = snapshot.child(userUsername).child("name").getValue(String.class);
                        String
phoneFromDB=snapshot.child(userUsername).child("phoneNo").getValue(String.class);
                        String emailFromDB = snapshot.child(userUsername).child("email").getValue(String.class);
                        String                    usernameFromDB                    =
snapshot.child(userUsername).child("username").getValue(String.class);
```

```
                Intent intent = new Intent(getApplicationContext(),MainActivity.class);
                SharedPreferences                    sharedPreferences=getSharedPreferences("userInfo",
Context.MODE_PRIVATE);
                SharedPreferences.Editor editor=sharedPreferences.edit();
                editor.putString("name",nameFromDB);
                editor.putString("phoneNo",phoneFromDB);
                editor.putString("email",emailFromDB);
                editor.putString("username",usernameFromDB);
                editor.putString("password",passwordFromDB);
                editor.putBoolean("isLoggedIn", true);
                editor.apply();
                startActivity(intent);
              } else {
                loginPassword.setError("Invalid Credentials");
                loginPassword.requestFocus();}
            } else {
              loginUsername.setError("User does not exist");
              loginUsername.requestFocus();}
        }
    }
    public void onBackPressed() {
      AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
      alertDialogBuilder.setTitle("Exit Application?");
      alertDialogBuilder
            .setMessage("Click yes to exit!")
            .setCancelable(false)
            .setPositiveButton("Yes",
                new DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int id) {
                    moveTaskToBack(true);
                    android.os.Process.killProcess(android.os.Process.myPid());
                    System.exit(1);}
                })
            .setNegativeButton("No", new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();}
            });
      AlertDialog alertDialog = alertDialogBuilder.create();
      alertDialog.show(); }
```

```
}
```

8.1.3 ALERT REMINDER ACTIVITY:

```
package com.example.myapplication;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import java.io.ByteArrayOutputStream;
public class alert_medicine extends AppCompatActivity {
    ImageView imageview1;
    public static String uriString = "";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_alert_medicine);
        Button button1 = findViewById(R.id.button1);
        Button button2 = findViewById(R.id.button2);
        imageview1 = findViewById(R.id.imageView);
        button1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               selectImage(); }
        });
        button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(alert_medicine.this, ResultActivity.class);
                intent.putExtra("uri", uriString);
                startActivity(intent); }
        });
    }
    private void selectImage() {
        final CharSequence[] options = {"Take Photo", "Choose from Gallery", "Cancel"};
        AlertDialog.Builder builder = new AlertDialog.Builder(alert_medicine.this);
        builder.setTitle("Add Photo!");
        builder.setItems(options, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int item) {
                if (options[item].equals("Take Photo")) {
                    Intent takePicture = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                    startActivityForResult(takePicture, 0);
                } else if (options[item].equals("Choose from Gallery")) {
                    Intent pickPhoto = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                    startActivityForResult(pickPhoto, 1);
                } else if (options[item].equals("Cancel")) {
                    dialog.dismiss();}
            }
        });
        builder.show(); }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
            case 0:
                if (resultCode == RESULT_OK && data != null) {
                    Bitmap selectedImage = (Bitmap) data.getExtras().get("data");
                    Uri selectedImageUri = getImageUri(alert_medicine.this, selectedImage);
                    Intent intent=new Intent(alert_medicine.this, CropperActivity.class);
                    intent.putExtra("DATA",selectedImageUri.toString());
                    startActivityForResult(intent,101); }
                break;
            case 1:
                if (resultCode == RESULT_OK && data != null) {
                    Uri selectedImageUri = data.getData();
                    Intent intent=new Intent(alert_medicine.this, CropperActivity.class);
                    intent.putExtra("DATA",selectedImageUri.toString());
                    startActivityForResult(intent,101);}
                break;
            case 101:
```

```
            if(resultCode==-1){
                String result=data.getStringExtra("RESULT");
                Uri selectedImageUri=null;
                if(result!=null){
                    selectedImageUri=Uri.parse(result);}
                imageview1.setImageURI(selectedImageUri);
                uriString=selectedImageUri.toString(); }
        }
    }
    public Uri getImageUri(Context inContext, Bitmap inImage) {
        ByteArrayOutputStream bytes = new ByteArrayOutputStream();
        inImage.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
        String path = MediaStore.Images.Media.insertImage(inContext.getContentResolver(), inImage, "Title",
null);
        return Uri.parse(path);
    }
    @Override
    public void onBackPressed() {
        super.onBackPressed();
        finish(); }
}
```

8.1.4 ResultActivity.class:

```
package com.example.myapplication;
import static android.icu.lang.UCharacter.toLowerCase;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.text.Html;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.mlkit.vision.common.InputImage;
import com.google.mlkit.vision.text.Text;
import com.google.mlkit.vision.text.TextRecognition;
import com.google.mlkit.vision.text.TextRecognizer;
import com.google.mlkit.vision.text.latin.TextRecognizerOptions;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;
public class ResultActivity extends AppCompatActivity {
    public static String med_name;
    public static String med_time="";
    public static String med_condition="";
    TextView textView;
    Button alarm_button;
    String DB_NAME = "Drugs_related_database.db";
    String TABLE_NAME = "drugs_for_common_treatments";
    DataBaseHelper myDBHelper;
    public static String uriStr="";
    static ArrayList<String> med_time_taken=new ArrayList<String>(Arrays.asList(med_time));
    static ArrayList<String> med_condition_detail=new ArrayList<String>(Arrays.asList(med_condition));
    public static String getValue() {
        return med_name;}
    @RequiresApi(api = Build.VERSION_CODES.N)
    public static List<String> getMedTime(){
        med_time_taken.removeIf(Objects::isNull);
        return med_time_taken;}
    @RequiresApi(api = Build.VERSION_CODES.N)
    public static List<String> getMedCondtion(){
        med_condition_detail.removeIf(Objects::isNull);
        return med_condition_detail; }
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_result);
        textView = findViewById(R.id.edittext1);
        alarm_button =findViewById(R.id.set_alarm_button);
       uriStr = getIntent().getStringExtra("uri");
        Uri uri = Uri.parse(uriStr);
        _extractTextFromUri(getApplicationContext(), uri);
        alarm_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(ResultActivity.this, SetAlarm.class)); }
        });
    }
    public void  _extractTextFromUri(Context context, Uri _uri) {
        TextRecognizer                                    recognizer                                    =
TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS);
        AssetDatabaseOpenHelper    assetDatabaseOpenHelper    =    new    AssetDatabaseOpenHelper(this,
DB_NAME);
        myDBHelper = new DataBaseHelper(this, DB_NAME);
        assetDatabaseOpenHelper.saveDatabase();
        try {
            InputImage image = InputImage.fromFilePath(context, _uri);
            Task<Text> result =
                    recognizer.process(image)
                            .addOnSuccessListener(new OnSuccessListener<Text>() {
                                @RequiresApi(api = Build.VERSION_CODES.N)
                                @SuppressLint("SetTextI18n")
                                @Override
                                public void onSuccess(Text visionText) {
                                    // Task completed successfully
                                    String[] lines = visionText.getText().split("\n");
                                    for (String line : lines) {
                                        String[] words = line.split(" ");
                                        StringBuilder buffer = new StringBuilder();
                                        for (String word : words) {
                                            int m = 1;
                                            Cursor res = myDBHelper.getAllData(TABLE_NAME, toLowerCase(word));
                                            if (res.getCount() != 0) {
                                                med_time_taken.clear();
                                                med_condition_detail.clear();
                                    while (res.moveToNext()) {
                                        buffer.append("    " + "<b><big> " + "<p style=" + "color:#143D38" +
">").append(m).append(".
").append(res.getString(0)).append("</p>").append("</big></b>").append("<br>");
                                        //buffer.append("---------- + <br>");
                                        buffer.append("<b>Medical Condition: </b>
").append(res.getString(1)).append("<br>");
                                        //buffer.append("<br> ");
                                        buffer.append("<b>Medicine Description: </b>
").append(res.getString(2)).append("<br>");
                                        buffer.append("<b>Medical URL: </b>
").append(res.getString(9)).append("<br>");
                                        buffer.append("<span style=" + "background-color:yellow" + ">" +
"<b>Time to take Medicine:  </b>").append(res.getString(11)).append("</span> <br>");
                                        String side_effect = res.getString(5);
                                        med_name = res.getString(0);
                                        med_time = res.getString(11);
                                        med_condition = res.getString(1);
                                        med_time_taken.add(med_time);
                                        med_condition_detail.add(med_condition);
                                        int a = res.getInt(0);
                                        if (a < 3) {
                                            buffer.append("<b> Drug Link:</b>
").append(res.getString(10)).append("</span> <br>");
                                        } else {
                                            buffer.append("<b>Drug Link:
</b>").append(res.getString(10)).append("<br>");}
                                        buffer.append("<br> ");
                                        buffer.append("<br>");
                                        m = m + 1; }
                                } else {
                                    continue; }
                            }
                            if (buffer.length() != 0) {
                                textView.setText(Html.fromHtml(buffer.toString()));
                            } else {
                                textView.setText("No Data Found");}
                        }}
                    })
                    .addOnFailureListener(
                            new OnFailureListener() {
                              @Override
                              public void onFailure(@NonNull Exception e) {
                                 // Task failed with an exception
       Toast.makeText(ResultActivity.this,"Textnotcaptured",Toast.LENGTH_SHORT).show();}
                          });
       } catch (IOException e) {
          e.printStackTrace();} }
}
```

8.1.5 SET ALARM.class :

```
package com.example.myapplication;
import android.Manifest;
import android.app.AlarmManager;
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.app.PendingIntent;
import android.app.TimePickerDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RadioButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import com.example.myapplication.adapter.DateCalculations;
import com.example.myapplication.database.AlarmDatabase;
import com.example.myapplication.database.MedicineDatabase;
import com.example.myapplication.models.AlarmModel;
import com.example.myapplication.models.MedicineModel;
import com.example.myapplication.recicver.AlarmReceiver;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
public class SetAlarm extends AppCompatActivity {
   int selectedposition;
   public SharedPreferences sharedPreferences;
   LinearLayout firstSlotLayout,specificDayWeek,DayIntervals;
   TextView medNameET,medTime,firstSlotTV,startDateTV;
   int id, numberOfSlot, noOfDays, daysInterval;
   EditText DayET,etDaysInterval;
   Spinner medCondtion;
   Calendar myCalender;
   String formattedTime;
   ImageView plusIV,mynasIV;
   CheckBox cbSunday,cbMonday,cbTuesday,cbWednesday,cbThursday,cbFriday,cbSaturday;
   String  medName,firstSlotTime, startDate, daysNameOfWeek, status, calculatedDate,
         newStartDate, medicineMeal, finalDate;
   boolean isEveryday, isSpecificDaysOfWeek, isDaysInterval;
   RadioButton  everydayRB,specificDayRB,daysIntervalRB,beforeMealRB,afterMealRB;
   boolean mon,sun,tue,wed,thu,fri,sat;
   boolean allPermission;
   MedicineDatabase dbHelper;
   String tableName = "";
```

```
int requestCode = 1;
int flag = 0;
int uniqueCode = 0;
int firstRequestCode;
Button nextPage;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_set_alarm);
    sharedPreferences = this.getSharedPreferences("alarmRequestCode", MODE_PRIVATE);
    requestCode = sharedPreferences.getInt("requestCodeValue", 1);
    flag = sharedPreferences.getInt("flagValue", 0);
    uniqueCode = sharedPreferences.getInt("uniqueCodeLastValue", 0);
    medNameET=findViewById(R.id.med_text);
    medTime=findViewById(R.id.med_time_text);
    medNameET.setText(ResultActivity.getValue());
    medCondtion=findViewById(R.id.no_of_times_SP);
    nextPage=findViewById(R.id.set_BTN);
    firstSlotTV=findViewById(R.id.first_slot_TV);
    firstSlotLayout=findViewById(R.id.first_slot_LAYOUT);
    beforeMealRB=findViewById(R.id.before_meal_RB);
    afterMealRB=findViewById(R.id.after_meal_RB);
    everydayRB=findViewById(R.id.everyday_RB);
    specificDayRB=findViewById(R.id.specific_day_RB);
    daysIntervalRB=findViewById(R.id.days_interval_RB);
    DayET=findViewById(R.id.no_of_days_ET);
    startDateTV=findViewById(R.id.start_date_TV);
    cbSunday=findViewById(R.id.cb_sunday);
    cbMonday=findViewById(R.id.cb_monday);
    cbTuesday=findViewById(R.id.cb_tuesday);
    cbWednesday=findViewById(R.id.cb_wednesday);
    cbThursday=findViewById(R.id.cb_thursday);
    cbFriday=findViewById(R.id.cb_friday);
    cbSaturday=findViewById(R.id.cb_saturday);
    specificDayWeek=findViewById(R.id.cv_specific_day_of_week);
    DayIntervals=findViewById(R.id.cv_days_interval);
    plusIV = findViewById(R.id.iv_plus);
    mynasIV = findViewById(R.id.iv_mynas);
    etDaysInterval = findViewById(R.id.et_days_interval);
```

```
everydayRB.setChecked(true);
beforeMealRB.setChecked(true);
ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
   adapter.addAll(ResultActivity.getMedCondtion());}
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
medCondtion.setAdapter(adapter);
medCondtion.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
   @RequiresApi(api = Build.VERSION_CODES.N)
   @Override
   public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
      if (position != -1) {
         String selectedCondition = (String) parent.getItemAtPosition(position);
         selectedposition = (ResultActivity.getMedCondtion()).indexOf(selectedCondition);
         String med_time_taken = (ResultActivity.getMedTime()).get(selectedposition);
         medTime.setText(med_time_taken);}
   }
});
firstSlotLayout.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      showHourPicker("Select first slot", 1);}
});
everydayRB.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
   @Override
   public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
      if (isChecked) {
         specificDayWeek.setVisibility(View.GONE);
         DayIntervals.setVisibility(View.GONE); }
   }
});
specificDayRB.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
   @Override
   public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
      if (isChecked) {
         specificDayWeek.setVisibility(View.VISIBLE);
         DayIntervals.setVisibility(View.GONE);}
   }
});
```

```
    daysIntervalRB.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
      @Override
      public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
          specificDayWeek.setVisibility(View.GONE);
          DayIntervals.setVisibility(View.VISIBLE); }
      }
    });
    startDateTV.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        showDatePicker();}
    });
    plusIV.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        int daysInterval = Integer.parseInt(etDaysInterval.getText().toString());
        daysInterval = daysInterval + 1;
        etDaysInterval.setText(String.valueOf(daysInterval));}
    });
    mynasIV.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        int daysInterval = Integer.parseInt(etDaysInterval.getText().toString());
        daysInterval = daysInterval - 1;
        if (daysInterval <= 1) {
          etDaysInterval.setText("1");
        } else {
          etDaysInterval.setText(String.valueOf(daysInterval)); }
      }
    });
    nextPage.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        if (Build.VERSION.SDK_INT >= 23 && !allPermission) {
          checkMultiplePermissions();}
else {
          mustExecute(); }
      }
    });
  }
  private String getMedicineMeal() {
    String meal = "";
    if (beforeMealRB.isChecked()) {
      meal = "Before Meal";
    } else if (afterMealRB.isChecked()) {
      meal = "After Meal";}
    return meal;}
  public static class DatePickerFragment extends DialogFragment implements
DatePickerDialog.OnDateSetListener {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
      final Calendar calendar = Calendar.getInstance();
      int year = calendar.get(Calendar.YEAR);
      int month = calendar.get(Calendar.MONTH);
      int day = calendar.get(Calendar.DAY_OF_MONTH);
      DatePickerDialog dpd = new DatePickerDialog(getActivity(),
          AlertDialog.THEME_HOLO_LIGHT, this, year, month, day);
      return dpd; }
    public void onDateSet(DatePicker view, int year, int month, int day) {
      TextView tv = (TextView) getActivity().findViewById(R.id.start_date_TV);
      Calendar cal = Calendar.getInstance();
      cal.setTimeInMillis(0);
      cal.set(year, month, day, 0, 0, 0);
      Date chosenDate = cal.getTime();
      DateFormat df_medium_uk = DateFormat.getDateInstance(DateFormat.MEDIUM, Locale.UK);
      String df_medium_uk_str = df_medium_uk.format(chosenDate);
      tv.setText(df_medium_uk_str); }
  }
  private void mustExecute() {
    if (checkValidity() && checkSpecificDayValidity()) {
      if (beforeMealRB.isChecked()) {
        tableName = "before_table";
      } else if (afterMealRB.isChecked()) {
        tableName = "after_table";}
      if (everydayRB.isChecked()) {
        String numberOfDays = DayET.getText().toString();
        DateCalculations dc = new DateCalculations();
```

```
id = 0;
medName = medNameET.getText().toString();
noOfDays = Integer.parseInt(numberOfDays);
isEveryday = true;
isSpecificDaysOfWeek = false;
isDaysInterval = false;
daysNameOfWeek = "null";
daysInterval = 0;
startDate = startDateTV.getText().toString();
newStartDate = startDate;
status = "not_taken";
medicineMeal = getMedicineMeal();
dbHelper = new MedicineDatabase(this);
getSlotTime();
calculatedDate = newStartDate;
for (int i = 0; i < noOfDays; i++) {
   MedicineModel medicineModel = new MedicineModel();
   medicineModel.setId(id);
   medicineModel.setDate(calculatedDate);
   medicineModel.setMedicineName(medName);
   medicineModel.setNumberOfSlot(numberOfSlot);
   medicineModel.setFirstSlotTime(firstSlotTime);
   medicineModel.setNumberOfDays(noOfDays);
   medicineModel.setEveryday(isEveryday);
   medicineModel.setSpecificDaysOfWeek(isSpecificDaysOfWeek);
   medicineModel.setDaysInterval(isDaysInterval);
   medicineModel.setDaysNameOfWeek(daysNameOfWeek);
   medicineModel.setDaysInterval(daysInterval);
   medicineModel.setStartDate(startDate);
   medicineModel.setStatus(status);
   medicineModel.setMedicineMeal(medicineMeal);
   medicineModel.setUniqueCode(uniqueCode);
   setAlarm(calculatedDate, firstSlotTime);
   dbHelper.insertData(medicineModel, tableName);
   uniqueCode++;
   SharedPreferences.Editor editor = sharedPreferences.edit();
   editor.putInt("uniqueCodeLastValue", uniqueCode);
   editor.commit();
   calculatedDate = dc.addDays(newStartDate, "1");
```

```
            newStartDate = calculatedDate;}
        } else if (specificDayRB.isChecked()) {
            String numberOfDays = DayET.getText().toString();
            DateCalculations dc = new DateCalculations();
            id = 0;
            medName = medNameET.getText().toString();
            noOfDays = Integer.parseInt(numberOfDays);
            isEveryday = true;
            isSpecificDaysOfWeek = false;
            isDaysInterval = false;
            daysNameOfWeek = "null";
            daysInterval = 0;
            startDate = startDateTV.getText().toString();
            newStartDate = startDate;
            status = "not_taken";
            medicineMeal = getMedicineMeal();
            dbHelper = new MedicineDatabase(this);
            getSlotTime();
            finalDate = startDate;
            for (int i = 0; i < noOfDays; i++) {
                calculatedDate = dc.addDays(newStartDate, "1");
                String singleDayName = dc.daysNameOfWeek(calculatedDate);
                if (daysNameOfWeek.contains(singleDayName)) {
                    MedicineModel medicineModel = new MedicineModel();
                    medicineModel.setId(id);
                    medicineModel.setDate(finalDate);
                    medicineModel.setMedicineName(medName);
                    medicineModel.setNumberOfSlot(numberOfSlot);
                    medicineModel.setFirstSlotTime(firstSlotTime);
                    medicineModel.setNumberOfDays(noOfDays);
                    medicineModel.setEveryday(isEveryday);
                    medicineModel.setSpecificDaysOfWeek(isSpecificDaysOfWeek);
                    medicineModel.setDaysInterval(isDaysInterval);
                    medicineModel.setDaysNameOfWeek(daysNameOfWeek);
                    medicineModel.setDaysInterval(daysInterval);
                    medicineModel.setStartDate(startDate);
                    medicineModel.setStatus(status);
                    medicineModel.setMedicineMeal(medicineMeal);
                    medicineModel.setUniqueCode(uniqueCode);
                    setAlarm(finalDate, firstSlotTime);
                    dbHelper.insertData(medicineModel, tableName);
                    uniqueCode++;
                    SharedPreferences.Editor editor = sharedPreferences.edit();
                    editor.putInt("uniqueCodeLastValue", uniqueCode);
                    editor.commit();
                    finalDate = calculatedDate;
                    newStartDate = calculatedDate;
                } else {
                    newStartDate = calculatedDate;
                    i--; }
            }
        } else if (daysIntervalRB.isChecked()) {
            String numberOfDays = DayET.getText().toString();
            DateCalculations dc = new DateCalculations();
            id = 0;
            medName = medNameET.getText().toString();
            noOfDays = Integer.parseInt(numberOfDays);
            isEveryday = true;
            isSpecificDaysOfWeek = false;
            isDaysInterval = false;
            daysNameOfWeek = "null";
            daysInterval = 0;
            startDate = startDateTV.getText().toString();
            newStartDate = startDate;
            status = "not_taken";
            medicineMeal = getMedicineMeal();
            dbHelper = new MedicineDatabase(this);
            getSlotTime();
            calculatedDate = startDate;
            for (int i = 0; i < noOfDays; i++) {
                MedicineModel medicineModel = new MedicineModel();
                medicineModel.setId(id);
                medicineModel.setDate(calculatedDate);
                medicineModel.setMedicineName(medName);
                medicineModel.setNumberOfSlot(numberOfSlot);
                medicineModel.setFirstSlotTime(firstSlotTime);
                medicineModel.setNumberOfDays(noOfDays);
                medicineModel.setEveryday(isEveryday);
```

```
                medicineModel.setSpecificDaysOfWeek(isSpecificDaysOfWeek);
                medicineModel.setDaysInterval(isDaysInterval);
                medicineModel.setDaysNameOfWeek(daysNameOfWeek);
                medicineModel.setDaysInterval(daysInterval);
                medicineModel.setStartDate(startDate);
                medicineModel.setStatus(status);
                medicineModel.setMedicineMeal(medicineMeal);
                medicineModel.setUniqueCode(uniqueCode);
                setAlarm(calculatedDate, firstSlotTime);
                dbHelper.insertData(medicineModel, tableName);
               uniqueCode++;
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putInt("uniqueCodeLastValue", uniqueCode);
                editor.commit();
              calculatedDate = dc.addDays(newStartDate, String.valueOf(daysInterval));
                newStartDate = calculatedDate;}
            }
            Toast.makeText(this, "Successfully added a medicine", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(this, AlarmView.class).putExtra("open", "medicine"));
            finish();
        } else {
            //  Toast.makeText(getContext(), "Please check all fields has been filled up", Toast.LENGTH_SHORT).show();
        }}
    private void setAlarm(String calculatedDate, String firstSlotTime) {
        String combine = calculatedDate + " " + firstSlotTime;
        setFinalAlarm(combine, 1);
        createAlarmModelObject();}
    private void setFinalAlarm(String combine, int value) {
        SimpleDateFormat sdf = new SimpleDateFormat("dd MMM yyyy hh:mm aaa");
        Calendar calendar = Calendar.getInstance();
        Calendar cal = Calendar.getInstance();
        try {
            calendar.setTime(sdf.parse(combine));
            int dateForAlarm = calendar.get(Calendar.DAY_OF_MONTH);
            int monthForAlarm = calendar.get(Calendar.MONTH);
            int yearForAlarm = calendar.get(Calendar.YEAR);
            int hourForAlarm = calendar.get(Calendar.HOUR_OF_DAY);
            int minuteForAlarm = calendar.get(Calendar.MINUTE);
            int secondForAlarm = 0;
            cal.set(yearForAlarm, monthForAlarm, dateForAlarm, hourForAlarm, minuteForAlarm,
secondForAlarm);
        } catch (ParseException e) {
            e.printStackTrace(); }
        AlarmManager alarmManager = (AlarmManager) this.getSystemService(Context.ALARM_SERVICE);
        Intent intent = new Intent(this, AlarmReceiver.class);
        intent.putExtra("medName", medName);
        intent.putExtra("mealStatus", medicineMeal);
        intent.putExtra("time", combine);
        intent.putExtra("status",status);
        intent.putExtra("med", "true");
        if (!firstSlotTime.equals("null") && value == 1) {
            firstRequestCode = requestCode;
            PendingIntent pendingIntent = PendingIntent.getBroadcast(this, firstRequestCode, intent,
PendingIntent.FLAG_IMMUTABLE);
            alarmManager.set(AlarmManager.RTC_WAKEUP, cal.getTimeInMillis(), pendingIntent);
            requestCode++; } else if (firstSlotTime.equals("null") && value == 1){
            firstRequestCode = -1;
            Log.d("entered", "entered 1"); }
    }
    private void createAlarmModelObject() {
        AlarmModel alarmModel = new AlarmModel();
        alarmModel.setId(0);
        alarmModel.setNdt(medName + uniqueCode);
        alarmModel.setNumberOfSlot(numberOfSlot);
        alarmModel.setFirstSlotTime(firstSlotTime);
        alarmModel.setFirstSlotRequestCode(firstRequestCode);
        AlarmDatabase alarmDatabase = new AlarmDatabase(this);
        alarmDatabase.insertAlarn(alarmModel);
        Log.d("firstRequest: ", ""+firstRequestCode);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putInt("requestCodeValue", requestCode);
        editor.commit(); }
    private void getSlotTime() {
        firstSlotTime = firstSlotTV.getText().toString();}
    private boolean checkValidity() {
        if (medNameET.getText().toString().equals("")) {
            medNameET.setError("Enter a medicine name");
            Toast.makeText(this, "Please enter a valid medicine name", Toast.LENGTH_SHORT).show();
            return false;
        } else if ((firstSlotTV.getText().toString().contains("Set"))) {
            if (firstSlotTV.getText().toString().contains("Set")) {
                Toast.makeText(this, "Please enter a valid time in slot 1", Toast.LENGTH_SHORT).show();}
            return false;
        } else if (DayET.getText().toString().equals("")) {
            DayET.setError("this field is required");
            Toast.makeText(this, "Number of days cannot be empty", Toast.LENGTH_SHORT).show();
            return false;
        } else if (startDateTV.getText().toString().contains("Touch here to set date")) {
            Toast.makeText(this, "Please enter a valid date", Toast.LENGTH_SHORT).show();
            return false;
        }else {
            return true; }
    }
    public void showDatePicker() {
        DialogFragment dFragment = new DatePickerFragment();
        dFragment.show(this.getFragmentManager(), "Date Picker");}
    public void showHourPicker(String message, final int number) {
        myCalender = Calendar.getInstance();
        int hour = myCalender.get(Calendar.HOUR_OF_DAY);
        int minute =myCalender.get(Calendar.MINUTE);
        TimePickerDialog.OnTimeSetListener myTimeListener = new TimePickerDialog.OnTimeSetListener()
{
            @Override
            public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                if (view.isShown()) {
                    myCalender.set(Calendar.HOUR_OF_DAY, hourOfDay);
                    myCalender.set(Calendar.MINUTE, minute);
                    try {
                        SimpleDateFormat sdf24 = new SimpleDateFormat("HH:mm");
                       // SimpleDateFormat sdf12 = new SimpleDateFormat("hh:mm a");
                        String strTime = myCalender.get(Calendar.HOUR_OF_DAY) + ":" +
myCalender.get(Calendar.MINUTE);
                        Date time = sdf24.parse(strTime);
                        formattedTime = sdf24.format(time);
                        firstSlotTV.setText(formattedTime);
                    } catch (ParseException e) {
```

```
                    e.printStackTrace(); }
                } }
        };
        TimePickerDialog timePickerDialog = new TimePickerDialog(this,
android.R.style.Theme_Holo_Light_Dialog_NoActionBar,myTimeListener,hour, minute,true);
        try {
            timePickerDialog.setTitle(message);
            timePickerDialog.getWindow().setBackgroundDrawableResource(android.R.color.transparent);
            timePickerDialog.show();
        } catch (Exception e) {
            Toast.makeText(this, e.getMessage(), Toast.LENGTH_SHORT).show();  }
    }
    public String getDaysNameOfWeek() {
        String daysName = "";
        if (cbSaturday.isChecked()) {
            daysName = daysName + "Saturday, ";
            sat = true; }
        if (cbSunday.isChecked()) {
            daysName = daysName + "Sunday, ";
            sun = true;}
        if (cbMonday.isChecked()) {
            daysName = daysName + "Monday, ";
            mon = true;}
        if (cbTuesday.isChecked()) {
            daysName = daysName + "Tuesday, ";
            tue = true;}
        if (cbWednesday.isChecked()) {
            daysName = daysName + "Wednesday, ";
            wed = true;}
        if (cbThursday.isChecked()) {
            daysName = daysName + "Thursday, ";
            thu = true;
        }
        if (cbFriday.isChecked()) {
            daysName = daysName + "Friday, ";
            fri = true;}
        return daysName.substring(0, daysName.length() - 2);
    }
    public boolean checkSpecificDayValidity() {
        if (specificDayWeek.getVisibility() == View.VISIBLE && cbSaturday.isChecked()) {
            return true;
        } else if (specificDayWeek.getVisibility() == View.VISIBLE && cbSunday.isChecked()) {
            return true;
        } else if (specificDayWeek.getVisibility() == View.VISIBLE && cbMonday.isChecked()) {
            return true;
        } else if (specificDayWeek.getVisibility() == View.VISIBLE && cbTuesday.isChecked()) {
            return true;
        } else if (specificDayWeek.getVisibility() == View.VISIBLE && cbWednesday.isChecked()) {
            return true;
        } else if (specificDayWeek.getVisibility() == View.VISIBLE && cbThursday.isChecked()) {
            return true;
        } else if (specificDayWeek.getVisibility() == View.VISIBLE && cbFriday.isChecked()) {
            return true;
        } else if (specificDayWeek.getVisibility() == View.GONE) {
            return true;
        } else {
 Toast.makeText(this, "Please select at least one specific day in week\nOr choose Everyday", Toast.LENGTH_LONG).show();
            return false;}
    }
    private void checkMultiplePermissions() {
        if (Build.VERSION.SDK_INT >= 23) {
            List<String> permissionsNeeded = new ArrayList<String>();
            List<String> permissionsList = new ArrayList<String>();
            if(!addPermission(permissionsList, android.Manifest.permission.WRITE_EXTERNAL_STORAGE)) {permissionsNeeded.add("Storage");
            }
            Map<String, Integer> perms = new HashMap<String, Integer>();
            perms.put(android.Manifest.permission.CAMERA, PackageManager.PERMISSION_GRANTED);
perms.put(android.Manifest.permission.WRITE_EXTERNAL_STORAGE,PackageManager.PERMISSION_GRANTED);
if (perms.get(Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED&& perms.get(android.Manifest.permission.WRITE_EXTERNAL_STORAGE)==PackageManager.PERMISSION_GRANTED) {
          allPermission = true;
            }
            if (permissionsList.size() > 0) {
                requestPermissions(permissionsList.toArray(new String[permissionsList.size()]),
                        StaticVariables.REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS);
            return;}
        }
    }
    private boolean addPermission(List<String> permissionsList, String permission) {
        if (Build.VERSION.SDK_INT >= 23)
            if (this.checkSelfPermission(permission) != PackageManager.PERMISSION_GRANTED) {
                permissionsList.add(permission);
                // Check for Rationale Option
                if (!shouldShowRequestPermissionRationale(permission))
                    return false;}
        return true;
    }
}
```

8.1.6 DetailsMedicineActivity.class:

```
package com.example.myapplication;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.os.Build;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupMenu;
import android.widget.TextView;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;
public class DetailsMedicineActivity extends AppCompatActivity {
    SharedPreferences sharedPreferences;
    Calendar calendar = Calendar.getInstance();
```

```
  TextView medNameTV, dateTimeTV,mealTV, numberOfSlotTV, firstSlotTV, numberOfDaysTV,
startDateTV, daysIntervalTV, statusTV,medicineDescription;
  ImageView  optionIV;
  LinearLayout statusLAYOUT;
  SendEmail sendMailTask;
  String medicineName,MedicineMeal, dateTime, numberOfSlot, firstSlotTime, numberOfDays, startDate,
daysInterval, status, imagePath, type,formattedPhoneNumber;
  Bitmap bitmap;
  String phoneUser,emailUser;
  @RequiresApi(api = Build.VERSION_CODES.N)
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_details_medicine);
    ActionBar actionBar = getSupportActionBar();
    if (actionBar != null) {
      actionBar.setDisplayHomeAsUpEnabled(true);}
    sharedPreferences=getSharedPreferences("userInfo", Context.MODE_PRIVATE);
     phoneUser = sharedPreferences.getString("phoneNo", "");
     emailUser = sharedPreferences.getString("email", "");
     formattedPhoneNumber = "+91" + phoneUser;
    init();
    receiveIntent();
    setTextAndIMageIntoView();
    setOnClick();
  }
  private void init() {
    medNameTV = (TextView) findViewById(R.id.medicine_name_TV_DA);
    dateTimeTV = (TextView) findViewById(R.id.date_and_time_TV_DA);
    numberOfSlotTV = (TextView) findViewById(R.id.number_of_slot_TV_DA);
    firstSlotTV = (TextView) findViewById(R.id.first_slot_TV_DA);
    numberOfDaysTV = (TextView) findViewById(R.id.number_of_days_TV_DA);
    startDateTV = (TextView) findViewById(R.id.start_date_TV_DA);
    daysIntervalTV = (TextView) findViewById(R.id.days_interval_TV_DA);
    statusTV = (TextView) findViewById(R.id.status_TV_DA);
    optionIV = (ImageView) findViewById(R.id.option_IV_DA);
    statusLAYOUT = (LinearLayout) findViewById(R.id.status_LAYOUT_DA);
    mealTV=(TextView) findViewById(R.id.mealTV);
  }
```

```
private void receiveIntent() {
    Intent intent = getIntent();
    medicineName = intent.getStringExtra("medName");
    dateTime = intent.getStringExtra("dateTime");
    numberOfSlot = intent.getStringExtra("numberOfSlot");
    firstSlotTime = intent.getStringExtra("firstSlotTime");
    numberOfDays = intent.getStringExtra("numberOfDays");
    startDate = intent.getStringExtra("startDate");
    daysInterval = intent.getStringExtra("daysInterval");
    status = intent.getStringExtra("status");
    type = intent.getStringExtra("type");
    MedicineMeal=intent.getStringExtra("medicineMeal");}
@RequiresApi(api = Build.VERSION_CODES.N)
private void setTextAndIMageIntoView() {
    medNameTV.setText(medicineName);
    dateTimeTV.setText(dateTime);
    numberOfSlotTV.setText(numberOfSlot);
    numberOfDaysTV.setText(numberOfDays);
    firstSlotTV.setText(firstSlotTime);
    startDateTV.setText(startDate);
    daysIntervalTV.setText(daysInterval);
    statusTV.setText(status);  // update the status text view
    mealTV.setText(MedicineMeal);
}
public void setOnClick() {
    optionIV.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            PopupMenu popupMenu = new PopupMenu(DetailsMedicineActivity.this, optionIV);
            popupMenu.getMenuInflater().inflate(R.menu.sms_email_option, popupMenu.getMenu());
            popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                @Override
                public boolean onMenuItemClick(MenuItem menuItem) {
                    switch (menuItem.getItemId()) {
                        case R.id.action_sms:
                            int numberOfDays1 = Integer.parseInt(numberOfDaysTV.getText().toString());
                                // Get the time of the first slot
                            String[] timeParts1 = firstSlotTime.split(":");
                            int hour1 = Integer.parseInt(timeParts1[0]);
                            int minute1 = Integer.parseInt(timeParts1[1]);
                                    // Set the first slot time in the calendar object
                            calendar.set(Calendar.HOUR_OF_DAY, hour1);
                            calendar.set(Calendar.MINUTE, minute1);
                           calendar.set(Calendar.SECOND, 0);
                            // Check if the current time is earlier than the desired time
                            if (calendar.getTimeInMillis() > System.currentTimeMillis()) {
                                // Schedule an alarm to send the SMS at the desired time
                                Intent intent = new Intent(getApplicationContext(), SMSSenderService.class);
                                intent.putExtra("phoneNumber", formattedPhoneNumber);
      intent.putExtra("message", "it's time to take " + medicineName + " at " + dateTime + " (" + status
     +","+MedicineMeal+ ")");
     PendingIntent pendingIntent = PendingIntent.getService(getApplicationContext(), 0, intent,
     PendingIntent.FLAG_IMMUTABLE);
     AlarmManageralarmManager=(AlarmManager)getApplicationContext().getSystemService(Context.ALAR
     M_SERVICE);
      alarmManager.set(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), pendingIntent);
                            } else {
      sendSMS(formattedPhoneNumber, "it's time to take " + medicineName + " at " + dateTime + " (" + status
     +","+MedicineMeal+ ")");
                            }
// Schedule email reminders for the selected number of days
     for (int i = 1; i < numberOfDays1; i++) {
                                calendar.add(Calendar.DAY_OF_MONTH, 1);
      sendSMS(formattedPhoneNumber, "it's time to take " + medicineName + " at " + dateTime + " (" + status
     +","+MedicineMeal+ ")"; }
                            return true;
                        case R.id.action_email:
                            // Get the number of days to schedule the reminders
                            int numberOfDays = Integer.parseInt(numberOfDaysTV.getText().toString());
                            // Get the time of the first slot
                            String[] timeParts = firstSlotTime.split(":");
                            int hour = Integer.parseInt(timeParts[0]);
                            int minute = Integer.parseInt(timeParts[1]);
                            // Set the first slot time in the calendar object
                            calendar.set(Calendar.HOUR_OF_DAY, hour);
                            calendar.set(Calendar.MINUTE, minute);
                            calendar.set(Calendar.SECOND, 0);
                            // Schedule email reminders for the selected number of days
                            for (int i = 0; i < numberOfDays; i++) {
```

```
                        calendar.add(Calendar.DAY_OF_MONTH, 1);
                        sendEmail();}
                    return true;
                default:
                    return false;
            }
        }
    });
    popupMenu.show();
  }
});
}
private void sendSMS(String formattedPhoneNumber, String s) {
    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(formattedPhoneNumber, null, s, null, null);
}
private void sendEmail() {
    // Get the recipient email, subject, and body
    String recipientEmail =emailUser;
    String subject = "Reminder of Medicine";
    String body = "it's time to take " +medicineName +" at " + getFormattedTime(calendar) + "and"+
status + " (" + MedicineMeal + ")";
    // Create a new SendEmailTask and execute it
    SendEmail task = new SendEmail(recipientEmail, subject, body);
    task.execute(); }
private String getFormattedTime(Calendar calendar) {
    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm", Locale.getDefault());
    return sdf.format(calendar.getTime()); }
@Override
public void onBackPressed() {
    super.onBackPressed();
    finish();}
}
```

8.1.7 Nearest_Hospital.class:

```
package com.example.myapplication;
import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.IntentSender;
import android.content.pm.PackageManager;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Looper;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.common.api.ResolvableApiException;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.google.android.gms.location.LocationSettingsResponse;
import com.google.android.gms.location.LocationSettingsStatusCodes;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
public class nearest_hospital extends AppCompatActivity implements OnMapReadyCallback {
   WebView webView;
   private LocationRequest locationRequest;
   @SuppressLint({"MissingInflatedId", "SetJavaScriptEnabled"})
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_nearest_hospital);
      locationRequest = LocationRequest.create();
      locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
      locationRequest.setInterval(5000);
      locationRequest.setFastestInterval(2000);
```

```
        webView = findViewById(R.id.webView2);
        webView.setWebViewClient(new WebViewClient());
        webView.getSettings().setJavaScriptEnabled(true);
        getCurrentLocation();
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == 1) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                if (isGPSEnabled()) {
                    getCurrentLocation();
                } else {
                    turnOnGPS();
                }
            }
        }
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 2) {
            if (resultCode == Activity.RESULT_OK) {
                getCurrentLocation();
            }
        }
    }
    private void getCurrentLocation() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if                                    (ActivityCompat.checkSelfPermission(nearest_hospital.this,
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
                if (isGPSEnabled()) {
                    LocationServices.getFusedLocationProviderClient(nearest_hospital.this)
                            .requestLocationUpdates(locationRequest, new LocationCallback() {
                                @Override
                                public void onLocationResult(@NonNull LocationResult locationResult) {
                                    super.onLocationResult(locationResult);
                                    LocationServices.getFusedLocationProviderClient(nearest_hospital.this)
                                .removeLocationUpdates(this);
                            if (locationResult != null && locationResult.getLocations().size() >0){
                                int index = locationResult.getLocations().size() - 1;
                                double latitude = locationResult.getLocations().get(index).getLatitude();
                                double longitude = locationResult.getLocations().get(index).getLongitude();

webView.loadUrl("https://www.google.co.in/maps/search/hospitals+near+me/@"+latitude+","+longitude+",
13.45z?hl=en&authuser=0");
                            }
                        }}, Looper.getMainLooper());
            } else {
                turnOnGPS(); }
        } else {
            requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);}
    }
  }
  private void turnOnGPS() {
      LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()
            .addLocationRequest(locationRequest);
      builder.setAlwaysShow(true);
      Task<LocationSettingsResponse> result = LocationServices.getSettingsClient(getApplicationContext())
            .checkLocationSettings(builder.build());
      result.addOnCompleteListener(new OnCompleteListener<LocationSettingsResponse>() {
         @Override
         public void onComplete(@NonNull Task<LocationSettingsResponse> task) {
            try {
               LocationSettingsResponse response = task.getResult(ApiException.class);
               Toast.makeText(nearest_hospital.this, "GPS is already tured on",
Toast.LENGTH_SHORT).show();
            } catch (ApiException e) {
               switch (e.getStatusCode()) {
                  case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:
                     try {
                        ResolvableApiException resolvableApiException = (ResolvableApiException) e;
                        resolvableApiException.startResolutionForResult(nearest_hospital.this, 2);
                     } catch (IntentSender.SendIntentException ex) {
                        ex.printStackTrace(); }
                     break;
                  case LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
                     //Device does not have location
```

```
                    break; }
              }
          } });
    }
    private boolean isGPSEnabled() {
        LocationManager locationManager = null;
        boolean isEnabled = false;
        if (locationManager == null) {
            locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE); }
        isEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
        return isEnabled;
    }}
```

8.2 XML Code :

8.2.1 Login.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:background="@drawable/gradient_6"
    tools:context=".LoginActivity">
    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        app:cardCornerRadius="30dp"
        app:cardElevation="20dp">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_gravity="center_horizontal"
            android:padding="24dp"
            android:background="@drawable/white_background">
            <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Login"
    android:textSize="36sp"
    android:textAlignment="center"
    android:textStyle="bold"
    android:textColor="@color/purple_700"/>
<EditText
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:id="@+id/login_username"
    android:background="@drawable/white_background"
    android:layout_marginTop="40dp"
    android:padding="8dp"
    android:hint="Username"
    android:drawableLeft="@drawable/ic_person"
    android:drawablePadding="8dp"
    android:textColor="@color/black"/>
<EditText
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:id="@+id/login_password"
    android:background="@drawable/white_background"
    android:layout_marginTop="20dp"
    android:padding="8dp"
    android:hint="Password"
    android:inputType="textPassword"
    android:drawableLeft="@drawable/ic_security"
    android:drawablePadding="8dp"
    android:textColor="@color/black"/>
<Button
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:text="Login"
    android:backgroundTint="@color/purple_700"
    android:id="@+id/login_button"
    android:textSize="18sp"
    android:layout_marginTop="30dp"
    app:cornerRadius = "20dp"/>
```

```
          <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/signupRedirectText"
            android:text="Not yet registered? Sign Up"
            android:layout_gravity="center"
            android:padding="8dp"
            android:layout_marginTop="10dp"
            android:textColor="@color/black"
            android:textSize="18sp"/>
      </LinearLayout>
   </androidx.cardview.widget.CardView>
</LinearLayout>
```

8.2.2 Alert Reminder Activity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#004"
   tools:context=".MainActivity">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Alert Reminder Activity"
      android:textColor="@color/yellow_shad1"
      android:textSize="30dp"
      app:layout_constraintBottom_toTopOf="@+id/button1"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.436"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
   <Button
      android:id="@+id/button1"
      android:layout_width="222dp"
      android:layout_height="70dp"
      android:layout_margin="10dp"
```

```
        android:layout_marginTop="140dp"
        android:text="PICK FILE"
        android:textSize="20sp"
        android:backgroundTint="#1de803"
        app:layout_constraintBottom_toTopOf="@+id/button2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.467"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.094" />
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="395dp"
        android:layout_height="332dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button1"
        app:layout_constraintVertical_bias="0.269"
        tools:srcCompat="@drawable/main_page_image" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:backgroundTint="#1de803"
        android:text="Details of Medicines"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.919" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

8.2.3 SetAlarm.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:background="@drawable/gradient_2"
android:orientation="vertical"
tools:context=".SetAlarm">
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:layout_marginBottom="55dp"
        android:background="@drawable/white_background"
        android:orientation="vertical"
        android:padding="15dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="ALARM SETTINGS"
            android:textAlignment="center"
            android:textColor="@color/lavender"
            android:textSize="36sp"
            android:textStyle="bold" />
        <!-- Medicine name-->
        <LinearLayout
            android:id="@+id/linearformedname"
            android:layout_width="match_parent"
            android:layout_height="100dp"
            android:layout_marginTop="20dp"
            android:orientation="vertical"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.0"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:ignore="MissingConstraints">
            <TextView
                android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Medicine Name"
            android:textAlignment="center"
            android:textColor="@color/colorPrimaryDark"
            android:textSize="20dp"
            tools:ignore="MissingConstraints" />
        <TextView
            android:id="@+id/med_text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:hint="Medicine_name"
            android:padding="20dp"
            android:textAlignment="center"
            android:textColor="@color/quantum_vanillared700"
            android:textSize="19sp"
            tools:ignore="MissingConstraints" />
    </LinearLayout>
    <!-- Medicine condtion-->
    <LinearLayout
        android:id="@+id/linearformedcondition"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearformedname"
        app:layout_constraintVertical_bias="0.078"
        tools:ignore="MissingConstraints">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="10dp"
            android:text="Medicine Condition"
            android:textAlignment="center"
            android:textColor="@color/colorPrimaryDark"
            android:textSize="20dp"
            tools:ignore="MissingConstraints" />
        <Spinner
            android:id="@+id/no_of_times_SP"
            android:layout_width="250dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:padding="15dp"
            android:textSize="15dp"></Spinner>
    </LinearLayout>
    <!--Medicine time taken-->
    <LinearLayout
        android:id="@+id/linearformedtime"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_marginTop="10dp"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearformedcondition"
        tools:ignore="MissingConstraints">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Medicine Time"
            android:textAlignment="center"
            android:textColor="@color/colorPrimaryDark"
            android:textSize="20dp"
            tools:ignore="MissingConstraints" />
        <TextView
            android:id="@+id/med_time_text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:hint="Medicine_time"
            android:padding="20dp"
            android:textAlignment="center"
            android:textColor="@color/quantum_vanillared700"
            android:textSize="19sp"
            tools:ignore="MissingConstraints"
            tools:layout_editor_absoluteX="25dp" />
    </LinearLayout>
    <!-- Time SLot to set alarm-->
    <LinearLayout
        android:id="@+id/first_slot_LAYOUT"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_margin="2dp"
        android:layout_weight="1"
        android:background="#f5f5f5"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearformedtime"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="2dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:gravity="center"
            android:padding="5dp"
            android:textSize="15dp"
            android:text="Select Slot 1" />
        <TextView
            android:id="@+id/first_slot_TV"
            android:layout_width="200dp"
            android:layout_height="0dp"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:background="@drawable/oval"
            android:padding="5dp"
            android:text="Set Time"
            android:textAlignment="center"
            android:textColor="#fff"
            android:textSize="16sp"
            app:cornerRadius="4mm" />
    </LinearLayout>
    <!-- Schedule the date-->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:orientation="vertical"
        android:padding="5dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Schedule"
            android:textSize="22sp" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:padding="5dp">
            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:padding="5dp"
                android:text="No of Days"
                android:textSize="15dp"
                />
            <EditText
                android:id="@+id/no_of_days_ET"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="2"
                android:importantForAutofill="no"
                android:inputType="number"
                android:maxLength="4"
                android:maxLines="1"
                android:padding="5dp"
                tools:ignore="LabelFor,SpeakableTextPresentCheck,TouchTargetSizeCheck" />
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal"
            android:padding="5dp">
            <TextView
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center_vertical"
                android:padding="5dp"
                android:text="Start Date"
                android:textSize="15dp"
                />
            <TextView
                android:id="@+id/start_date_TV"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:padding="5dp"
                android:text="Touch here to set date"
                android:textColor="@color/colorPrimary"
                android:textSize="18sp" />
        </LinearLayout>
    </LinearLayout>
    <!-- Days -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:orientation="vertical"
        android:padding="5dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Days"
            android:textSize="22sp" />
        <RadioGroup
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <RadioButton
                android:id="@+id/everyday_RB"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="5dp"
                android:text="Everyday"
                android:textSize="15dp"/>
            <RadioButton
                android:id="@+id/specific_day_RB"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="5dp"
                android:text="Specific day of week"
                android:textSize="15dp"/>
            <RadioButton
                android:id="@+id/days_interval_RB"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="5dp"
                android:text="Days interval"
                android:textSize="15dp"/>
        </RadioGroup>
    </LinearLayout>
    <!--Specific Day of week-->
    <LinearLayout
        android:id="@+id/cv_specific_day_of_week"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:orientation="vertical"
        android:padding="5dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Select specific day of week"
            android:textSize="15dp"/>
        <CheckBox
            android:id="@+id/cb_sunday"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Sunday"
            android:textSize="16sp" />
        <CheckBox
            android:id="@+id/cb_monday"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Monday"
            android:textSize="16sp" />
        <CheckBox
            android:id="@+id/cb_tuesday"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Tuesday"
            android:textSize="16sp" />
        <CheckBox
            android:id="@+id/cb_wednesday"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Wednesday"
            android:textSize="16sp" />
        <CheckBox
            android:id="@+id/cb_thursday"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Thursday"
            android:textSize="16sp" />
        <CheckBox
            android:id="@+id/cb_friday"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Friday"
            android:textSize="16sp" />
        <CheckBox
            android:id="@+id/cb_saturday"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Saturday"
            android:textSize="16sp" />
    </LinearLayout>
    <!--Interval of days-->
    <LinearLayout
        android:id="@+id/cv_days_interval"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:orientation="vertical"
        android:padding="5dp">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Select days interval"
            android:textSize="15dp"/>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:orientation="horizontal">
            <ImageView
                android:id="@+id/iv_mynas"
                android:layout_width="50dp"
                android:layout_height="50dp"
                android:background="@drawable/round_shape"
                android:clickable="true"
                android:focusable="true"
            android:padding="8dp"
            app:srcCompat="@drawable/ic_mynus"
            tools:ignore="VectorDrawableCompat" />
        <EditText
            android:id="@+id/et_days_interval"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:gravity="center"
            android:inputType="number"
            android:padding="5dp"
            android:text="1" />
        <ImageView
            android:id="@+id/iv_plus"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:background="@drawable/round_shape"
            android:clickable="true"
            android:focusable="true"
            android:padding="8dp"
            app:srcCompat="@drawable/ic_plus"
            tools:ignore="VectorDrawableCompat" />
    </LinearLayout>
</LinearLayout>
<!-- Selecting the medicine (Before Meal or After Meal)  -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:orientation="vertical"
    android:padding="5dp">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="8dp"
        android:paddingBottom="8dp"
        android:text="Select when medicine should take?"
        android:textSize="16sp" />
    <RadioGroup
        android:layout_width="match_parent"
```

```
                android:layout_height="wrap_content"
                android:orientation="horizontal">
                <RadioButton
                    android:id="@+id/before_meal_RB"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:text="Before Meal"
                    android:textSize="15dp"/>
                <RadioButton
                    android:id="@+id/after_meal_RB"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:text="After Meal"
                    android:textSize="15dp"/>
            </RadioGroup>
        </LinearLayout>
        <Button
            android:id="@+id/set_BTN"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:background="@drawable/oval"
            android:text="Save"
            android:textColor="#fff"
            android:textSize="16sp" />
    </LinearLayout></ScrollView></LinearLayout>
```

8.2.4 Nearest_hospital.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".nearest_hospital">
    <WebView
        android:id="@+id/webView2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

CHAPTER 9

RESULTS

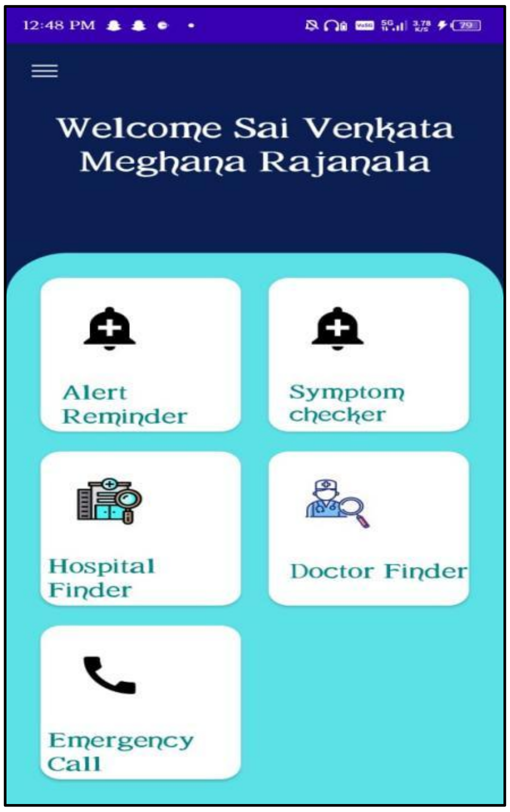


Figure 9.1 Main Page of Medminder App

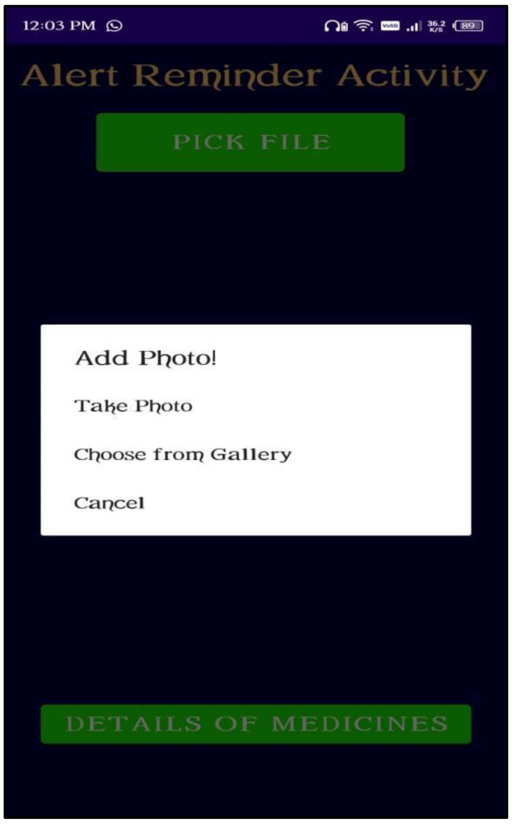
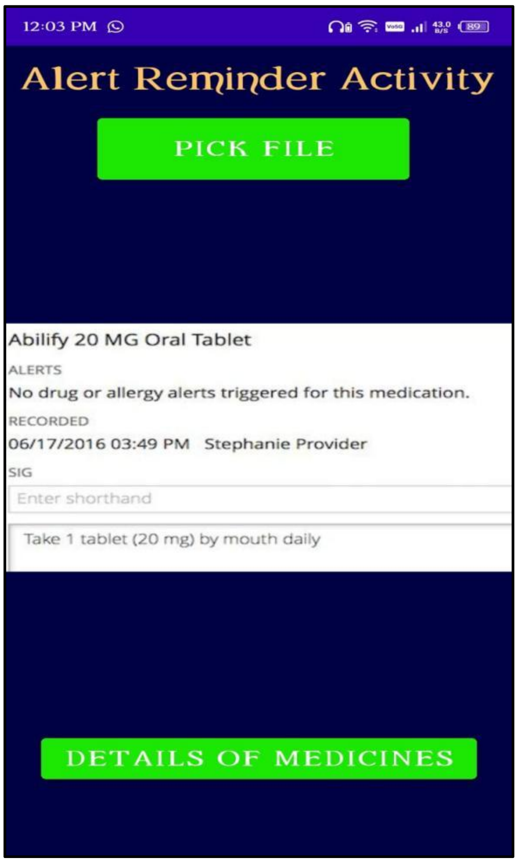


Figure 9.2 Alert Reminder Activity to upload Prescript



Abilify 20 MG Oral Tablet

ALERTS

No drug or allergy alerts triggered for this medication.

RECORDED

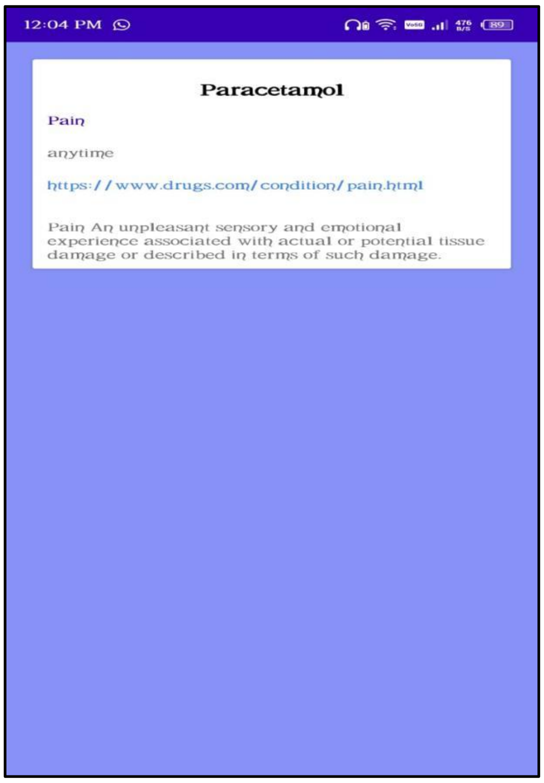
06/17/2016 03:49 PM Stephanie Provider

SIG

Enter shorthand

Take 1 tablet (20 mg) by mouth daily

Figure 9.3 Alert Reminder Activity after uploading prescription



Paracetamol

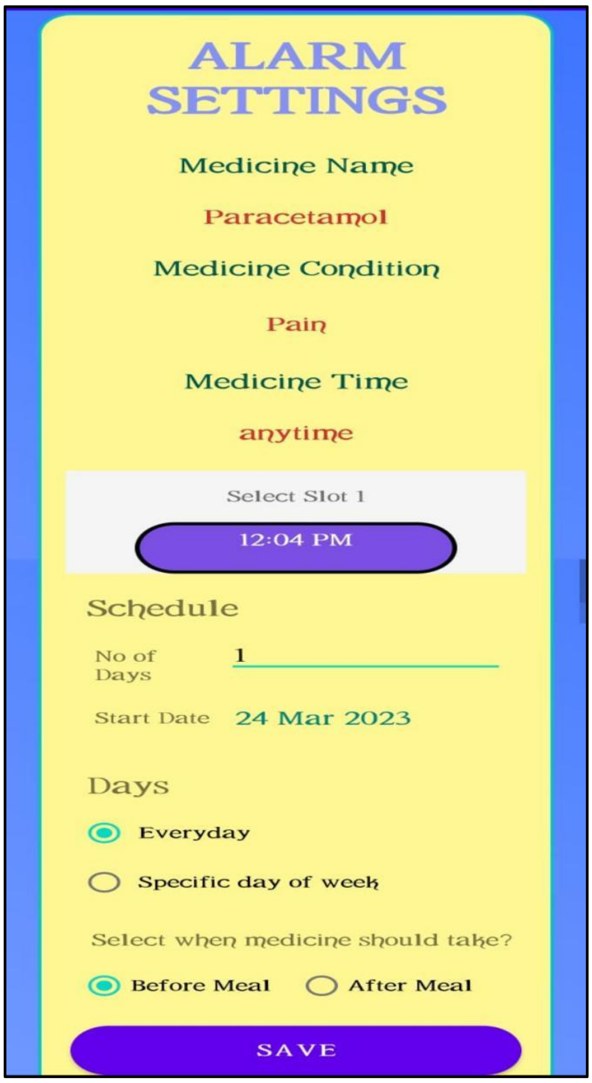
Pain

anytime

https://www.drugs.com/condition/pain.html

Pain An unpleasant sensory and emotional experience associated with actual or potential tissue damage or described in terms of such damage.

Figure 9.4 Details of Medicines extracted from prescription



ALARM SETTINGS

Medicine Name

Paracetamol

Medicine Condition

Pain

Medicine Time

anytime

Select Slot 1

12:04 PM

Schedule

No of Days: 1

Start Date: 24 Mar 2023

Days

[✓ Everyday]

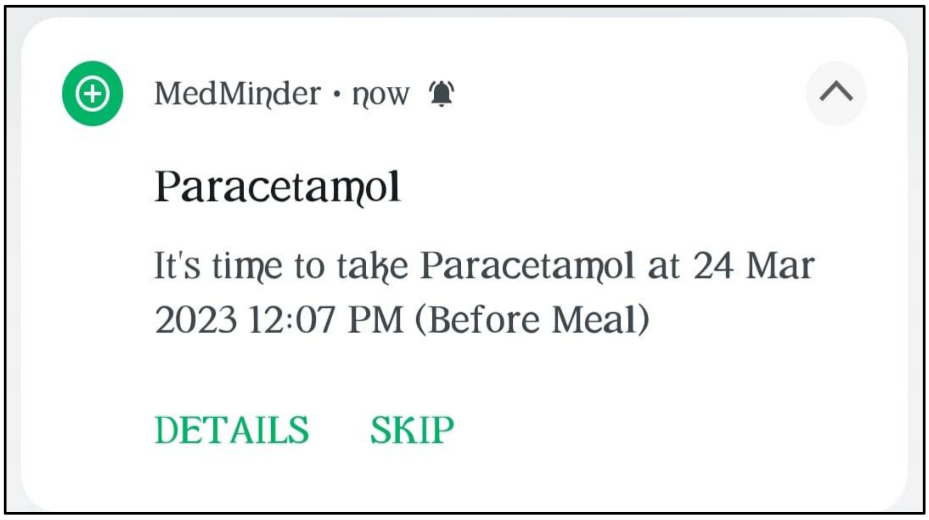
Specific day of week

Select when medicine should take?

[✓ Before Meal] After Meal

SAVE

Figure 9.5 Alarm Fixing based on Medicine details



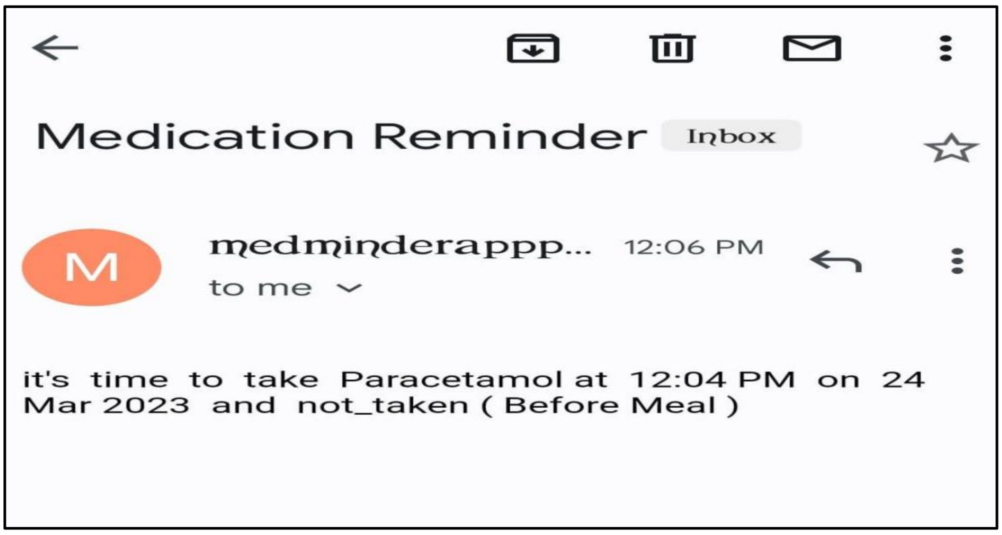
MedMinder • now

Paracetamol

It's time to take Paracetamol at 24 Mar 2023 12:07 PM (Before Meal)

DETAILS SKIP

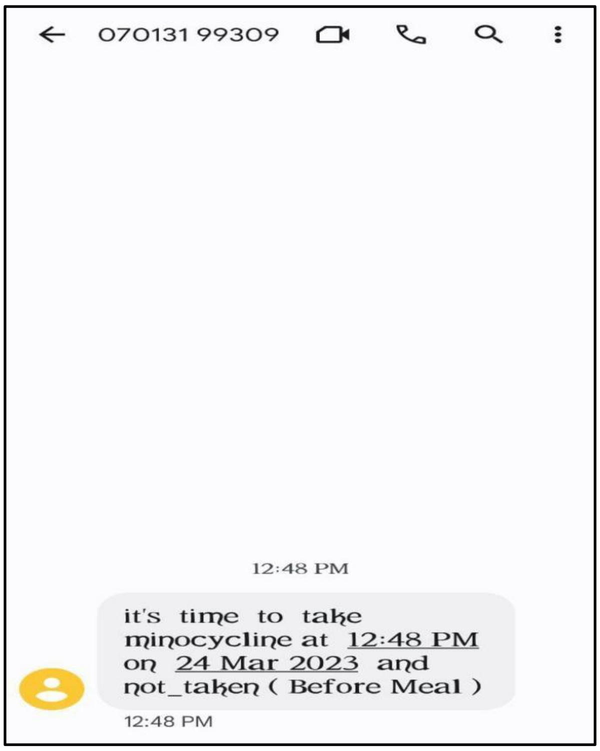
Figure 9.6 Medicine Reminder Notification


Medication Reminder Inbox

medminderappp... 12:06 PM
to me

it's time to take Paracetamol at 12:04 PM on 24 Mar 2023 and not_taken (Before Meal)

Figure 9.7 Medicine Email Reminder Notification at set time


07013199309

12:48 PM

it's time to take minocycline at 12:48 PM on 24 Mar 2023 and not_taken (Before Meal)

12:48 PM

Figure 9.8 Medicine SMS Reminder Notification at set time

CHAPTER 10

FUTURE IMPLEMENTATION

In near future, We can add handwritten prescription recognition and we can add a potential voice recognition where it would allow users to speak the name of their medication, and the app would automatically set the alarm for them. And use natural language processing which enable app to understand user text input and set alarm accordingly.

CHAPTER 11

CONCLUSION

MedMinder is an Android app designed to help users manage their medications and monitor their symptoms. The app features include medication reminders, symptom tracking, hospital and doctor finders, and emergency messaging.

MedMinder is simple and user-friendly interface makes it easy for users to manage their medications based on their doctor's prescription. The app ability to send notifications to emergency contacts ensures that users receive prompt assistance in case of an emergency.

REFERENCE

1. **Tabi K., Randhawa A.S., Choi F., Mithani Z., Albers F., Schnieder M., Nikoo M., Vigo D., Jang K., Demlova R. and Krausz M.,(2019).** “Mobile Apps for Medication Management: Review and Analysis” JMIR Mhealth Uhealth.
2. **Firebase”. Available at en.m.wikipedia.org/wiki/Firebase.[Visited December 17, 2019.]**
3. **B. Moysset, C. Kermorvant, and C. Wolf, “Full-page text recognition”:** Learning where to start and when to stop, arXiv preprint arXiv: 1704, 08628, 2017
4. **Slagle, J.M., Gordon, J.S., Harris, C.E., Davison, C.L., Culpepper, D.K., Scott P. and Johnson, K.B., (2011)** “MediHealth – Designing a next generation system for child-centered medication management”, Journal of Biomedical Informatics, Vol. 43, No. 5, pp. 27-31.
5. **The Mobile Revolution**: How Mobile Technologies Drive a Trillion-Dollar Impact (2015,January).
6. **Becker, E., Metsis, V., Arora, R., Vinjumur, J.K., Xu, Y. and Makedon, F. (2009)** “SmartDrawer: RFID- Based smart medicine drawer for assistive environments”, Proc. of Pervasive technologies related to assistive environments, June, pp 1-8.
7. **Ammouri, S. and Bilodeau, G.A. (2008)** “Face and hands detection and tracking applied to the monitoring of medication intake”, Proc. of Canadian Conf. on Computer and Robot Vision, May, pp. 147-154.
8. **Prasad, B., (2013) “Social media, health care, and social networking”**, Gastrointest Endosc. Vol. 77, pp 492–495.
9. **Batz, D., Batz, M., Lobo, N.D.V. and Shah, M. (2005)** “A computer vision system for monitoring medication intake”, Canadian Conf. on Computer and Robot Vision, May, pp. 362-369.
10. **Zao, J.K., Wang, M.Y., Peihsuan, T. and Liu, J.W.S., (2010) "Smart Phone Based Medicine Intake Scheduler, Reminder and Monitor"**, IEEE e-Health Networking Applications and Services (Healthcom), pp 162 – 168
11. **Rosaly B. Alday and Ruel M. Pagayon MediPic**: A Mobile Application for Medical Prescriptions in 2015

IJIEMR Transactions, online available on 21st Feb 2023. Link :http://www.ijiemr.org/downloads.php?vol=Volume-12&issue=Issue 03

10.48047/IJIEMR/V12/ISSUE 03/25

Title ALERT MESSAGE USING DOCTOR'S PRESCRIPTION

Volume 12, ISSUE 03, Pages: 184-188

Paper Authors
Rajesh.Y, Sai Venkata Meghana.R, Aparna Sri. M, Zehra Banu. SD

International Journal for Innovative Engineering and Management Research

[illegible]

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Alert Message Using Doctor's Prescription

Rajesh.Y[1], Assistant Professor, Department of Computer Science and Engineering, Andhra Loyola Institute of Engineering and Technology, Vijayawada.
rajeshrajiv1324@gmail.com
Sai Venkata Meghana.R[2], IV B.Tech Department of Computer Science and Engineering, Andhra Loyola Institute of Engineering and Technology, Vijayawada.
Aparna Sri. M[3], IV B.Tech Department of Computer Science and Engineering, Andhra Loyola Institute of Engineering and Technology, Vijayawada.
Zehra Banu. SD[4], IV B.Tech Department of Computer Science and Engineering, Andhra Loyola Institute of Engineering and Technology, Vijayawada.

Abstract
This paper presents the development of Alert Message using Doctor's Prescription- "MEDMINDER", an android application designed to help patients manage their medication schedules and track their symptoms. The app allows users to take picture of their prescription, and using optical character recognition (OCR) technology, the app extracts relevant tablet details and sets medication reminders. Additionally, the app allows users to receive SMS or Email reminders based on their preferences. The app's front end was implemented using Android studio and java, and the back-end was developed using Firebase and SQLite DB Browser. The app runs on android devices such as smartphones and tablet. In addition to medication reminders and symptom checking, the app allows users to find hospitals and doctors based on their location and updates their personal profile.

Keywords*:* OCR, Android, Reminder System, Notification system, symptoms checker.

Introduction

Health and patients are two important topics that are closely related. Health refers to the overall well-being of an individual, including their physical, mental, and emotional state. Patients are individuals who are receiving medical treatment for an illness or health condition.

When it comes to patient care, health is a critical factor to consider. Patients who are in good health generally have a better prognosis and may respond more positively to treatment. On the other hand, patients who are in poor health may have a more challenging recovery process and may be at higher risk of complications.

Health and patients are also linked in terms of prevention. Maintaining good health can help to prevent the development of many illnesses and health conditions, which can ultimately reduce the need for medical treatment. For example, individuals who eat a healthy diet, exercise regularly, and avoid smoking and excessive alcohol consumption may be less likely to develop chronic conditions such as heart disease, diabetes, and certain types of cancer.

In terms of patient care, prevention is also important. Health care providers may work with patients to develop strategies for maintaining good health and preventing the development of future health problems. This may include lifestyle changes, such as dietary modifications and exercise, as well as regular check-ups and screenings to identify potential health concerns early on.

Overall, health and patients are two interconnected concepts that are essential to the provision of high-quality medical care. By focusing on both health promotion and patient-centered care, health care providers can help patients to achieve better health outcomes and improve their overall quality of life.

Literature Review

[1] PAPER: Development of an Android Based Medication Reminder and Adherence System
AUTHORS: Adeyemi, T.O.
ANALYSIS: The proposed system is useful to patients to set a customized alarm tone

in their local language or select from a list of default tones and allows specialists to automatically see the list of patients connected to them and their chat messages. And assist patients with chronic illness like Cancer, Diabetes, Asthma and HIV/AIDS, to get notifications from medical personnel about the availability of drugs and also served as a reminder system.

[2] PAPER: Medication Reminder and HealthCare Application
AUTHORS: Deepti Ameta, Kalpana Mudaliar and Palak Patel
ANALYSIS: The proposed system is based on Android Operating system which will remind the users to take medicines on time through notification and automatic alarm ringing system. The users will get the schedule of medicine in-take time with medicine description, starting and ending date of medicine, notification through message or email, automatic alarm ringing system and navigation system.

[3] PAPER: Medical Handwritten Prescription Recognition Using CRNN
AUTHORS: Rayan Haidar, Roger Achkar
ANALYSIS: The proposed system established a Convolutional Recurrent Neural Network (CRNN) technology using Python that can interpret handwritten English prescriptions and translate them into digital text. For this, datasets with 66 different classes, including alphanumeric characters, punctuation, and spaces, were used. Since prescriptions generally contain two or three words, the training was carried out using short texts. Normal handwriting and prescriptions from doctors were used to train the model. The system got a 98% accuracy rate after taking training time and data input into account.

Proposed System

People most of them facing difficulty in taking on time and unable to understand medicines. So, we want to propose an android application which can be used for setting the alarm from prescriptions itself and give the notifications by various ways such as Sound, Messages via through email or normal message.

By combining the android with OCR (Optical Character Recognition). we going to develop an app which gives alert for the patients with medicine prescription and we can also include another module like predicting the disease by using symptoms from the patient.

Technology Description

Android:

Android is a mobile operating system that was developed by Google in 2008. It is based on the Linux kernel and is designed to run on touchscreen mobile devices such as smartphones and tablets. Android has become one of the most popular operating systems for mobile devices worldwide, with over 2 billion active users as of 2021.

One of the key features of Android is its open-source nature. This means that the source code is freely available to anyone who wants to use, modify, or distribute it. This has led to a vibrant community of developers and enthusiasts who have created numerous innovative and useful applications for android devices.

Android also includes several built-in features and applications that make it a powerful and versatile platform. These include Google Maps, Google Assistant, and Google Play Services, which provide users with access to a wide range of services and tools.

One of the benefits of using Android is its integration with other Google services such as Gmail, Google Drive, and Google Calendar. This makes it easy for users to access their data and files from anywhere, and to keep their devices in sync with their other devices and accounts.

Firebase:

Firebase is a mobile and web application development platform that was developed by Google. It provides a wide range of tools and services to help developers build, test, and deploy mobile and web applications quickly and easily. Firebase includes features such as real-time database, authentication, cloud messaging, and analytics, among others.
Firebase also provides a range of authentication options, including email and password, phone number, and social media logins. This makes it easy for developers to add secure user authentication to their applications

without having to build the functionality from scratch.

SQLite Database:
SQLite is a popular open-source relational database management system that is used in a variety of applications, including mobile devices, web browsers, and desktop software. It is lightweight, fast, and highly reliable, making it an ideal choice for developers who need a simple and efficient way to store and manage data. One of the key benefits of SQLite is its ease of use. Unlike other database management systems that require a separate server process to be running, SQLite is a self-contained library that can be embedded directly into an application. This means that developers can easily create and manage SQLite databases without the need for additional setup or configuration.
SQLite databases can be created using a variety of programming languages, including C, C++, Java, Python, and others. The SQLite library provides a simple set of APIs that can be used to create tables, insert data, query data, and perform other operations on the database.

Implementation
Image Uploading:
The Image Uploading feature is implemented using the Android Camera and Gallery. When the user clicks on the Alert Reminder button, the app opens the

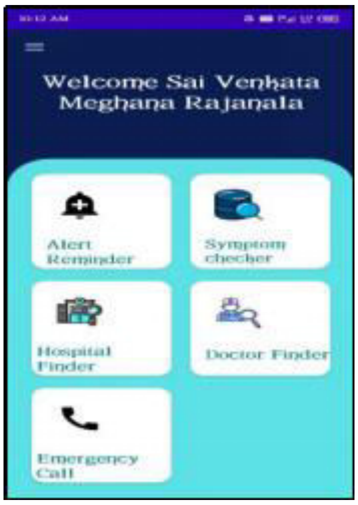


Fig 1: Main Page of App

camera or gallery to allow the user to select an image. Once the user selects an image, the app processes the image to extract medicine details.

Medicine Details Extraction:

The Medicine Details Extraction feature is implemented using the Google Vision API. The API is used to extract the text from the prescription image and parse it to extract medicine details. The medicine name, medicine conditions, and time are extracted and stored in the app database.

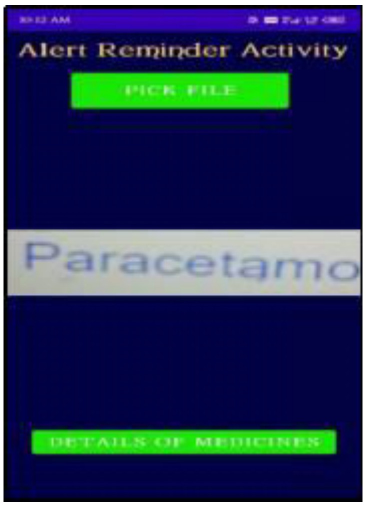


Fig 2 Alert Reminder Page of App

Alarm Setting:

The Alarm Setting feature is implemented using the Android Alarm Manager class. When the user uploads a prescription image, the app sets an alarm for the specific time that the medicine needs to be taken. The alarm is set to repeat daily, so the user will receive a reminder every day at the same time. When the alarm triggers, the app displays a notification reminding the user to take their medicine.

Alarm View Activity:

The Alarm View Activity displays a list of all the alarms set by the user. The list is displayed in a Recycler View, and each item in the list displays the medicine name, time, and conditions. The user can also edit or delete the alarm from this view.

Symptom Checker:

The Symptom Checker feature is implemented using a Recycler View that displays a list of symptoms. When the user clicks on a symptom, the app displays information about the diseases associated with that symptom. The information is retrieved from a local database that contains information about various diseases and their symptoms.

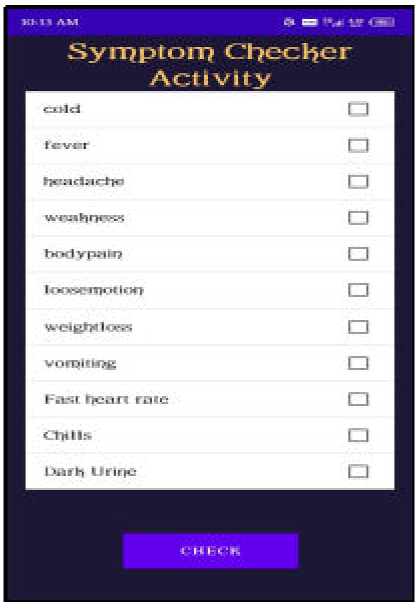

Symptom Checker Activity
cold ☐
fever ☐
headache ☐
weakness ☐
bodypain ☐
loosemotion ☐
weightloss ☐
vomiting ☐
Fast heart rate ☐
Chills ☐
Dark Urine ☐
CHECK

Fig 3 Symptom Checking Page of App

Emergency Message:

The Emergency Message feature is implemented using the Android SMS Manager class. When the user clicks on the Alert button, the app sends an SMS message to all the contacts in the user added contacts. The message contains a URL that, when clicked, opens a map showing the user's current location

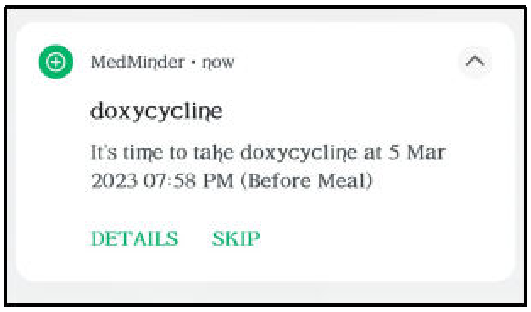

MedMinder • now
doxycycline
It's time to take doxycycline at 5 Mar 2023 07:58 PM (Before Meal)
DETAILS SKIP

Fig 4 Notification to remind Medicine

The alarm is set to ring without any ringtone music to avoid any distraction.

SMS and Email Notification:

The SMS and Email Notification feature is implemented using the Android SMS Manager and Java Mail APIs.

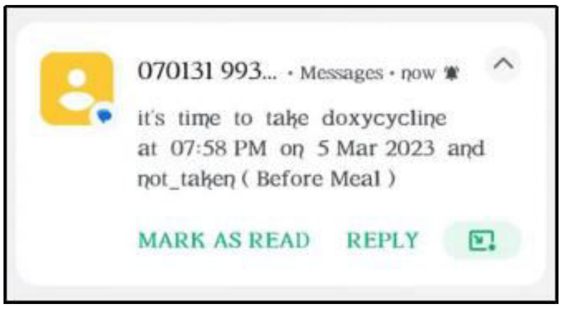

070131 993... • Messages • now
it's time to take doxycycline at 07:58 PM on 5 Mar 2023 and not_taken (Before Meal)
MARK AS READ REPLY

Fig 5 SMS for remind Medicine

When the alarm rings, the app sends an SMS and email notification to the user's emergency contacts. The SMS and email notification include the medicine name, medicine conditions, and time when it should be taken.

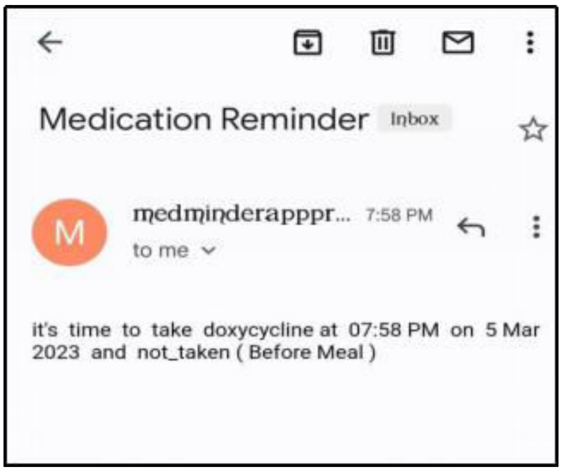

Medication Reminder Inbox
medminderapppr... 7:58 PM
to me
it's time to take doxycycline at 07:58 PM on 5 Mar 2023 and not_taken (Before Meal)

Fig 6 Email notification for remind Medicine

Future Implements
In near future, we can add a potential voice recognition where it would allow users to speak the name of their medication, and the app would automatically set the alarm for them. And use natural language processing which enable app to understand user text input and set alarm accordingly.

Conclusion

MedMinder is an Android app designed to help users manage their medications and monitor their symptoms. The app features include medication reminders, symptom tracking, hospital and doctor finders, and emergency messaging. MedMinder is simple and user-friendly interface makes it easy for users to manage their medications based on their doctor's prescription. The app ability to send notifications to emergency contacts ensures that users receive prompt assistance in case of an emergency.

References

[1] T O Adeyemi & Elizabeth Amusan (2020, September). Development of an Android Based Medication Reminder and Adherence System.

[2] Chua, J. Y., Koh, D., Lim, H. Y., & Tan, J.Y. (2019, December). An Android application for medication reminder and management.

[3] Roger Achkar; Khodor Ghayad; Rayan Haidar; Sawsan Saleh; Rana Al Hajj (2019, October). Medical Handwritten Prescription Recognition Using CRNN.