PROJECT REPORT

ON

MEDICINE REMINDER ANDROID APP

Submitted by

1.Vaishnavi Milind Rathi
2.Bharti Sureshrao Pokale
3.Vaishnavi Suresh Choudhari

Guided by
Prof. Monika Sharma

Department of Computer Application
BCA

VIDYA BHARATI MAHAVIDYALAYA,

C. K. NAIDU ROAD, CAMP,
AMRAVATI-444 602.

2020-2021

CERTIFICATE

This is to certify that the Project Report on

"Medicine Reminder Android App"

Has been submitted by

1.Vaishnavi Milind Rathi

2.Bharti Sureshrao Pokale

3.Vaishnavi Suresh Choudhari

Of B.C.A. Third Year (Semester – VI) in a satisfactory manner under my supervision and guidance in the partial fulfilment of the Subject Mini Project for the degree of

Bachelor of Computer Application

Sant Gadge Baba Amravati University, Amravati. During the academic year **2020-2021**

Prof. Monika Sharma **Project Guide**	External Examiner	Prof. Vinod Mohod **Head of the Department**

Dr. Pradnya S. Yenkar
Principal
Department of Computer Application (BCA)
VIDYA BHARATI MAHAVIDYALAYA,
C. K. NAIDU ROAD, CAMP,
AMRAVATI - 444 602.
2020-2021

ACKNOWLEDGEMENT

We would like to thank the project guide **Prof. Monika Sharma**, for encouraging throughout the project and course duration. It is great pleasure for us to acknowledge the assistance and contributions of our Head of the Department **Prof. Vinod N. Mohod**, for his prompt and timely help in the official clearances and valuable suggestions during the development of this project.

We are thankful for the constant encouragement given by our project guide and rest of the faculties and other staff of my department.

We also owe our thanks to non-teaching staff whose unfailing invaluable help from time to time made the completion of this project a reality and thanks to all those whom we have not been able to thanks individually.

Last but not least, we express our heartiest gratitude to our Parents for their love and blessings to complete the project successfully.

1.Vaishnavi Milind Rathi

2.Bharti Sureshrao Pokale

3.Vaishnavi Suresh Choudhari

BCA-III (Sem-VI)

Table of Contents

1. Abstract

This is an Android-based application in which an automatic alarm ringing system is implemented. It focuses on doctor and patient interaction. Patients need not remember their medicine dosage timings as they can set an alarm on their dosage timings. The alarm can be set for multiple medicines and timings including date, time and medicine description.

Good health has been a major concern since the inception of mankind whilst for some people attaining good health requires taking prescribed medicines or pills routinely. However, many patients find it very difficult to keep track of taking their medication in the right time and proportion. This happens especially if it involves taking pills or medication on daily basis due to several reasons such as heavy work load, forgetfulness, old age and alterations in day-today behaviour can also have a significant result on whether patients recall to take their prescribed medications which can be termed as medicine adherence, which is a very serious problem because it may affect the total well-being of the patient, delaying the curing time, raising the total medical cost of the patient and can be a matter of life and death. The aim of the study is to design and develop an automated Android based mobile application for medicine or pill reminder as prescribed by a doctor to patients.

Our system takes up the prescription details from the user such as the duration of the prescription, the names of the medicines, the times they are to be taken and the amount of each medicine which is to be taken. After all this data has been entered, our system will remind the user at the prescribed time of which medicine is to be taken in form of a mobile notification. The patients can leave taking medicines to just our app. Whenever the time for the medicine is up, they will be notified and they only have to take their prescriptions during that time,

and no other time. If implemented properly, this will drastically decrease overdose of medicines due to forgetfulness and the patients will also be reminded to take their medicines.

2. Introduction

Medication adherence is a growing concern throughout the healthcare industry with doctors, healthcare systems, and other stakeholders (insurance companies) since the elderly or senior patients' medication has a big issue of drugs misuse. It is very likely for them to forget to take their pills on time. Especially, those who take multiple medications at the same time. Also, they might take wrong dosage accidentally which may lead to unfortunate consequences such as death. This is a clear proof that it is a widespread problem and clearly related to adverse patient outcomes and higher healthcare costs.

Many medical errors are due to the fact that people in charge of patient or elder's medication have to deal with sorting huge amounts of pills each day. This paper consists on the conception, design and working of Android app to remind people to take their medicine on time.

The designed Android based medicine reminder will be of great help for patients suffering from range of problems such as forgetfulness, busy schedules, old age, cognitive disorders, bad working conditions, Alzheimer disease, loss of memory, dementia, people with emotional problems, stress, anxiety, depression, and also individuals with a very hectic work schedules or lifestyle.

2.1 Project Objective

To develop system that keeps track of and guides people to effectively provide drugs to patients as prescribed and reminds them at the time of taking the dozes.

i. To study the current system and review literature related to the system to be developed.

ii. To remind patients about taking their prescribed medicines or pills within the stipulated time as prescribed by a doctor.

iii. To use an alarm ringing system in making the remembrance in order to make patients stay healthy and fit.

iv. To test and validate the developed system to ensure that it operates as originally specified.

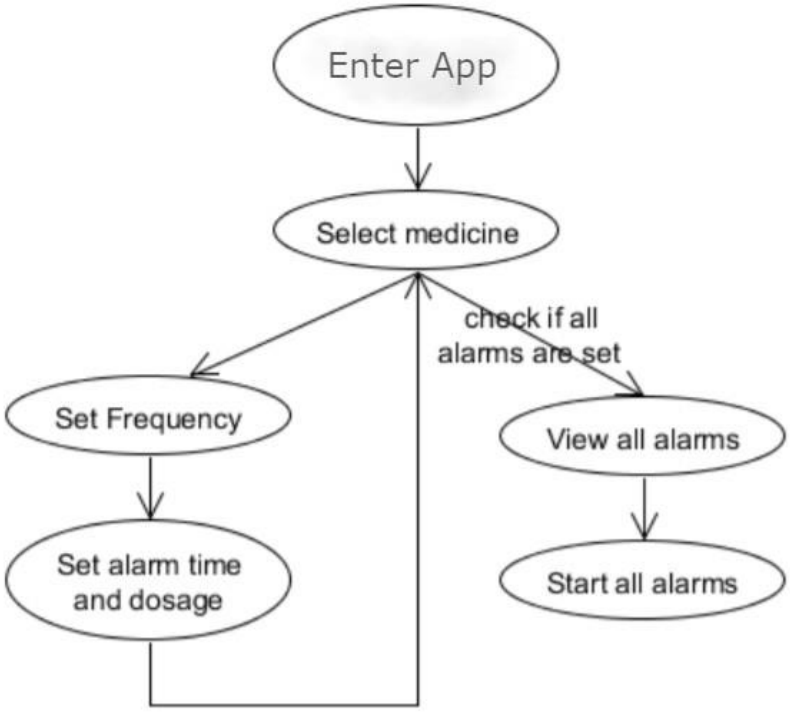
2.2 Project Overview

This study aim to design and develop an automated Android based application for medicine or pill reminder as prescribed by a doctor to patients using the Java programming language and the Android Studio integrated development environment.

Participants allocated to the medication reminder app group will have access to a medication reminder app in addition to standard care. The medication reminder app has a basic feature, which is the ability to provide simple daily reminders with no interactivity to reinforce correct medication-taking behaviour. The app reminders will act similarly to an alarm to remind the patients when it is time to take their medications. The app provides a one-time only reminder at each scheduled time, with no ability to snooze or reschedule the reminder. There is

also no ability to register whether the medication was taken or not, once the reminder alerts the patient to take the medication.

Fig. Displays the flow of steps taken by the patients to set an alarm



2.3 Study of System

The study demonstrates how to design, develop and implement the Android based medication or pill reminder mobile application by making an alarm to the patients, using Android Studio integrated development environment.

• My application will provide some important health related services to the users.

• It is so much user friendly and all the services are kept in one single activity.

• All the features are clearly visible, user won't have any problem using our application.

• Saving user's valuable times and money.

• Saves its user from being harass.

• User can get medicine alarm.

• The designed application would help patients to maximize the full benefit of the medicine and abstaining them from the risk that result as not taking the medicine or pill within the stipulated time and proportion as prescribed by the specialist.

3. Problem Identification

Before the development work begins, requirements elicitation was carried out by distributing requirements questionnaire to potential users. The requirements questionnaire lists out questions about the application requirements, organized by proposed functionalities in the application. The objectives of the requirements questionnaire are as follows:

• To find out whether the respondents are interested in medication reminder application.

• To find out whether the respondents are using the similar application.

• To find out how respondents think about the importance of using a medicine reminder application.

• To find out whether the respondents can recognize the type of medicine.

• To find out the respondent is a chronically ill patient or otherwise in order to assess whether they need the application.

• To know whether the respondent has a regular medical check-up in order to assess the need of appointment functions.

• To design features that are important and useful for users.

• To find out the majority type of users, whether novice, intermediate or expert users.

• To figure out what respondent think about the benefits after using this application. The results from the requirements questionnaire show that 85% of respondents who are not in good health conditions would use the proposed application. 80% of them do not has any medication reminder mechanism to help them manage their medicines. 40% of the respondents strongly believe that the application is necessary because 70% of the respondents are not able to recognize different types of medicines. In addition, 63.2% of the respondents do not have a regular medical check-up. The remaining 36.8% who has the routine, perform their medical check-up once a year (75%) or once every six months (25%).

4. Proposed Work

In helping the public to manage their medication, this paper proposes yet another medicine reminder application called medicine apk, which is based on alarm system. Smartphone is the closest device to a person, and a medication alarm does not appear intrusive to the smartphone users. The alarm has to be set in advance and may be customized following the patient's preference.

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. Android is a Linux-based operating system designed primarily for touch screen mobile devices such as smart phones and tablet computers, developed by Google in conjunction with the Open Handset Alliance.

Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. The system is specified on android operating system only because the

market share of Android is high. [9] Android also comes with an application development framework (ADF), which provides an API for application development and includes services for building GUI applications, data access, and other component types. The framework is designed to simplify the reuse and integration of components. Android apps are built using a mandatory XML manifest file. The manifest file values are bound to the application at compile time. This file provides essential information to an Android platform for managing the life cycle of an application. Examples of the kinds of information included in a manifest file are descriptions of the app's components among other architectural and configuration properties. Components can be one of the following types: Activities, Services, Broadcast Receivers, and Content Providers.

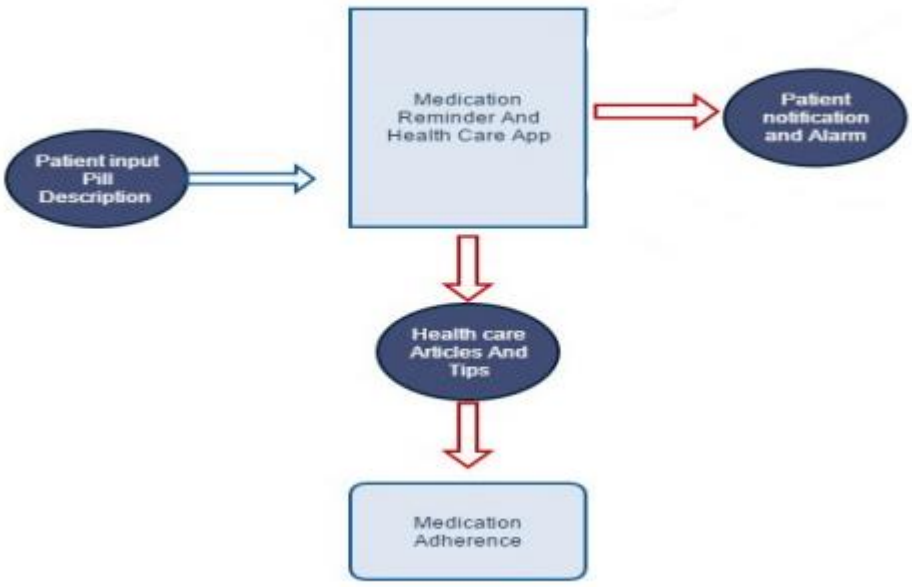


Figure 1. Medication Reminder and Healthcare: System Overview

Medication reminders help in decreasing medication dispensing errors and wrong dosages. The Reminder system consists of two parts –setting Alarm and getting notification. Set Alarm module- It helps in reminding about the medicines.

User can add details of his dosage schedules. Using the date field one can enter the starting and ending dates between which, he has to take medicines.

The time field shows the time of dosage and on that time the alarm will get rung. The user can add the description of the medicine, including name, purpose and other related description. All the information will be saved in the database. This makes any time availability of the patients' records. They can change the ringtone of the alarm from the ringtones stored in the devices.

Get Notification module: Once the alarm is set then the user gets the notification. The users can activate or deactivate this accordingly. If he does not require the notification he can turn off it. If he requires this system then a notification will be sent into his device.

5. Survey of Technologies

This application is built on Android platform which is an open-source technology that has greatly inspired developers to generate plenty of applications. Android is a software stack for mobile devices that includes an operating system based on Linux platform in the lower level, middleware which consists of libraries, android runtime environment and the required application framework and key applications such as Contacts, Browser, and Message as shown in Figure 18. Android applications are written in Java programming language executed using a custom virtual machine (VM) called Dalvik created by Google. This Dalvik VM is responsible for converting and executing Java byte code. The Android Software Development Kit (SDK) provides the tools and APIs necessary to start developing applications on the Android platform. Android Studio

is the official integrated development environment (IDE) for Android application development, based on IntelliJ IDEA. It is available for download on Windows, Mac OS X and Linux. Android Studio has many new features when compared to that of Eclipse Android Development Tool Kit (ADT). These features include Gradle, Cloud integration, lint and Dynamic layout view. Android Studio embeds a powerful layout editor for the users to drag and drop controls into the view and preview on many different devices at the same time as shown in Figure.

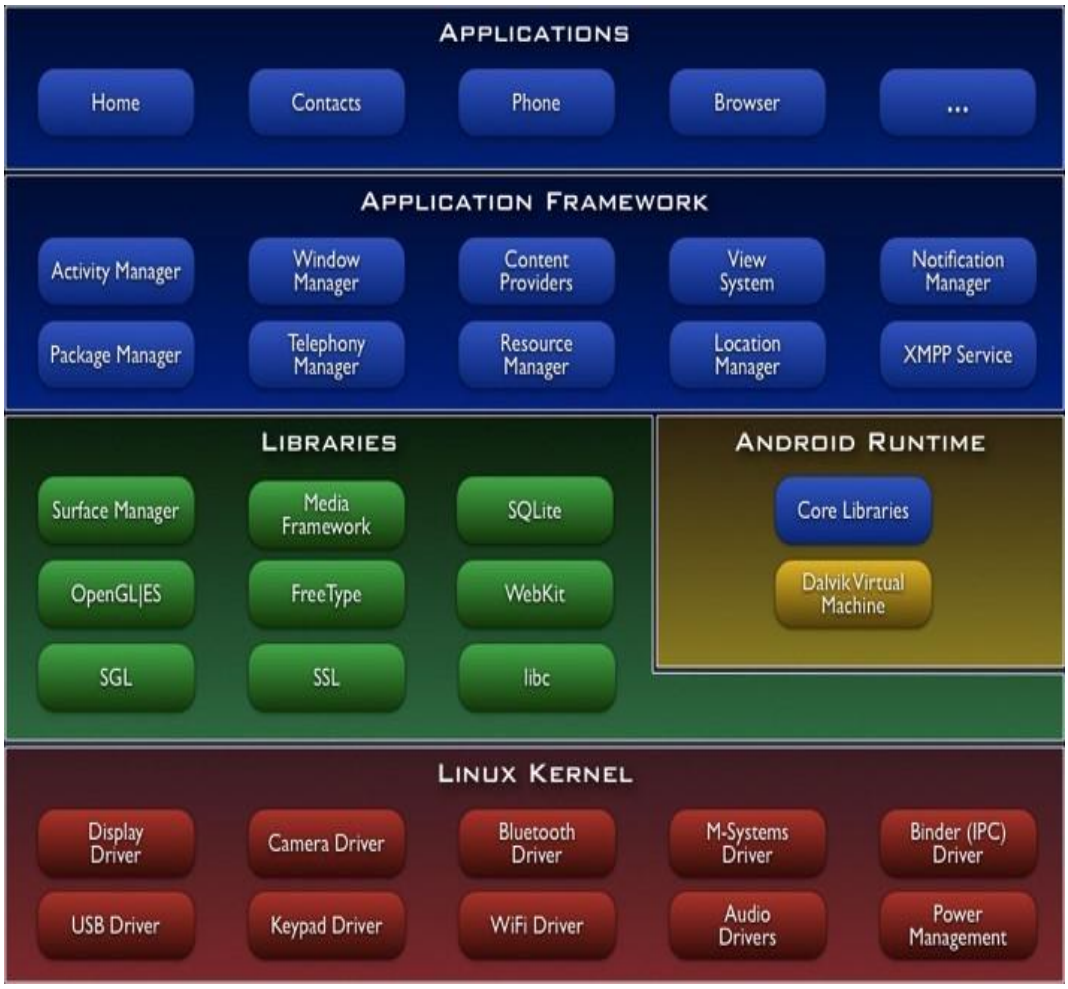


Figure : Android architecture

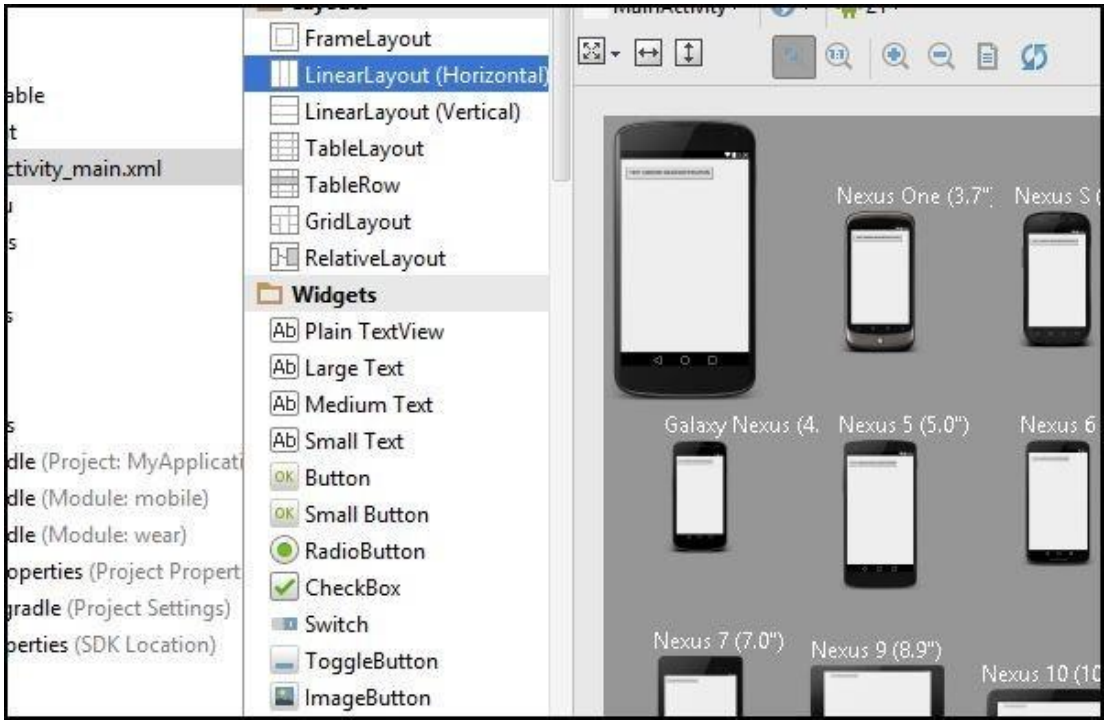


Figure : Android Studio Layout Editor

The *medicine* application is implemented using Android Studio 1.1 on a Windows 8.1 64-bit machine along with the Java Platform, Standard Edition, JDK 7 version. The target platform for this application is 21, Android 5.0 Lollipop version. For displaying all the tracking information of the patient in a pdf file an external library called Droidtext is used. Droidtext is an open-source external library that can be added to an android project to create, format and download a pdf file. In Android Studio, an application can be modelled as a package of components, each of which can be instantiated and run as necessary. These components could be activities, services, broadcast receivers and content providers. Activity forms the basis of the user interface. Service runs in the background and remains active even if windows are switched. Broadcast Receiver reacts asynchronously to messages from other applications and Content Provider stores data relevant to the application, usually in a database which could be shared across applications.

6. Front End Tools

6.1 Extensive Markup Language

What is XML?

XML stands for extensible markup language. A markup language is a set of codes, or tags, that describes the text in a digital document. The most famous markup language is hypertext markup language (HTML), which is used to format Web pages. XML, a more flexible cousin of HTML, makes it possible to conduct complex business over the Internet.

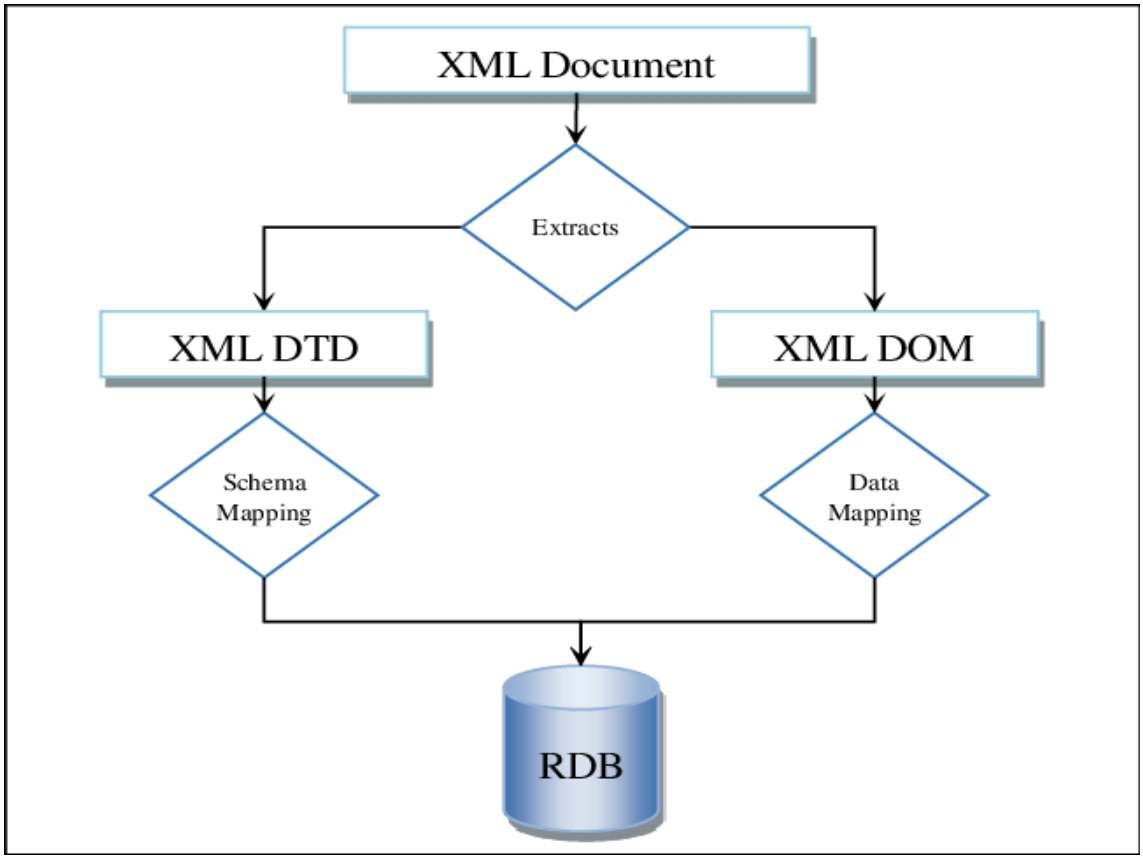
What are XML's advantages over HTML?

Whereas HTML tells a browser application how a document should look, XML describes what's in the document. In other words, XML is concerned with how information is organized, not how it is displayed. (XML formatting is done through separate style sheets.)

To illustrate, consider the following HTML tags: the command signals a paragraph and word translates into **word**. The HTML tags are fixed; every site developer uses the same tags to do the same things. XML, by contrast, lets you create your own tags to label the meaning or use of data. So if you're using XML to describe a widget you're selling, your tags might look like this: $100" SKU="555432" dealer="Widgets Incorporated">.

XML's flexibility has many benefits. It lets you transfer data among corporate databases and Web sites without losing crucial descriptive information. It lets you automatically customize the presentation of data rather than display

the same page to all comers. And it makes searches more efficient because search engines can sort through precise tags rather than long pages of text.



6.2 Android (Java)

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA and is designed specifically for Android Development. It is to Android like Eclipse is to Java. This can be downloaded on Windows, macOS and Linux. This IDE provides many useful features to make Android development flexible for developers. It is a very user-friendly IDE as one does not require an android phone to run and test the application.

Emulator is provided with it to stimulate a real-time device to run and debug the applications in Android Studio itself with the help of AVD manager. When an application is developed on Android Studio, it will be automatically installed on the emulator to run and debug which is faster than running on an android device.

It is a gradle-based system which runs as an integrated tool from the Android Studio menu.

It provides many features which can be achieved without modifying the app's core source files. Features include customizing, configuring and extending the build process, can create multiple APKS for the app with different features done with the same project and modules and lastly, the code and resources can be reused. The dependencies of the project are also specified in the build.gradle file which are made available in the build by the gradle.

It is easy and flexible to design a layout for the application being developed as it has the feature of dragging and dropping the UI components which automatically generate the components for it in the xml files. It has instant run option which makes it easier to push all the changes made in the application to the running app without having to rebuild a new APK file for that. Android project view contains all the project files. Each app module will contain manifests, java and res files. The manifest files contain the AndroidManifest.xml file, java files contain the java source code files along with the Junit test code and res code consists of all the xml layouts, UI strings and bitmap images.

Java is the technology of choice for building applications using managed code that can execute on mobile devices.

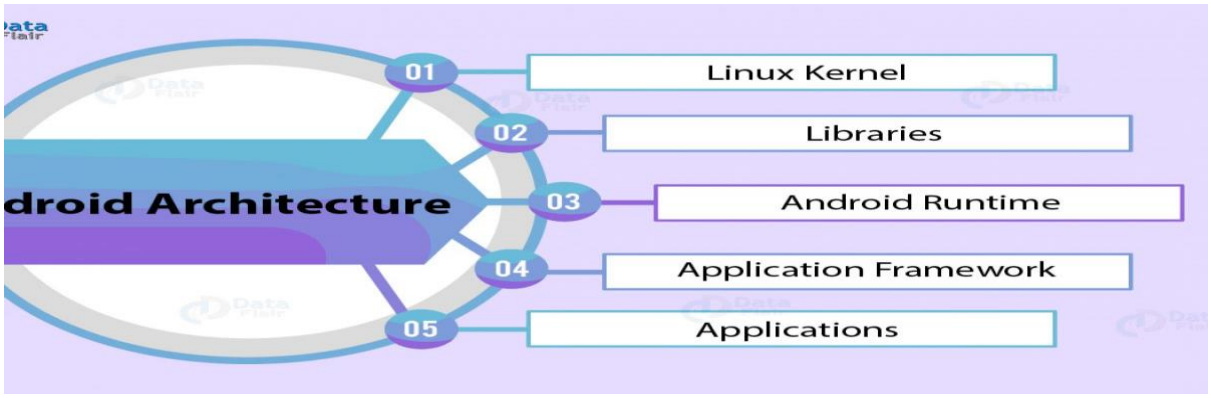
Android is an open source software platform and Linux-based operating system for mobile devices. The Android platform allows developers to write managed code using Java to manage and control the Android device. Android applications can be developed by using the Java programming language and the Android SDK. So, familiarity with the basics of the Java programming language is a prerequisite for programming on the Android platform. This article discusses where Java fits in mobile application development and how we can use Java and Android SDK to write applications that can work on Android devices.

The Choice of Java

What made Java be the technology of choice for mobile development for the Android platform? The Java Programming Language emerged in the mid-1990s; it was created by James Gosling of Sun Microsystems. Incidentally, Sun Microsystems was since bought by Oracle. Java has been widely popular the world over, primarily because of a vast array of features it provides. Java's promise of "Write once and run anywhere" was one of the major factors for the success of Java over the past few decades.

Java even made inroads into embedded processors technology as well; the Java Mobile Edition was built for creating applications that can run on mobile devices. All these, added to Java's meteoric rise, were the prime factors that attributed to the decision of adopting Java as the primary development language for building applications that run on Android. Java programs are secure because they run within a sandbox environment. Programs written in Java are compiled into intermediate code known as bytecode. This bytecode is then executed inside the context of the Java Virtual Machine. You can learn more about Java from this link.

Using Java for Building Mobile Applications



The mobile edition of Java is called Java ME. Java ME is based on Java SE and is supported by most smartphones and tablets. The Java Platform Micro Edition (Java ME) provides a flexible, secure environment for building and executing applications that are targeted at embedded and mobile devices. The applications that are built using Java ME are portable, secure, and can take advantage of the native capabilities of the device. Java ME addresses the constraints that are involved in building applications that are targeted at mobile devices. In essence, Java ME addresses the challenge of executing applications on devices that are low on available memory, display, and power.

There are various ways to build applications for Android devices, but the recommended approach is to leverage the Java programming language and the Android SDK. You can explore more about the Android SDK Manager from here.

To get started using Java for Android, you should first download and install Android Studio. You then may want to take advantage of the SDK Manager to download and install the latest SDK tools and platforms.

7. Backend Tools

This application uses SQLite primarily for storage and some details are stored in shared preferences in key-value pairs.

SQLite

SQLite is a relational database management system. It is not a client-server database, but is embedded into the end program. It is a popular choice to be used an embedded database software for local/client storage in application software. SQLite library becomes an important part of the application, and its functionality is used by the application through simple function calls that can reduce the latency in database access. It stores the entire database which includes tables, definitions, indices, and the data itself as a single cross-platform file on the machine.

As it is a server-less design, less configuration is required by SQLite applications than client-server databases. Here, several processes may not be able to write to the database file due to its design. For simple queries with little concurrency, SQLite performance profits from avoiding the overhead of passing its data to another process. It is a high-reliability storage solution and has been used without problems in billions of smart phones around the world.

SHARED PREFERENCES

Shared Preferences allow you to save and retrieve data in the form of key, value pair. The first parameter is the key, and the second parameter is the mode. In order to use shared preferences, you have to call a method getSharedPreferences() that

returns a SharedPreference instance pointing to the file that contains the values of preferences. One can save data in shared preferences using SharedPreferences.Editor class.

8. Software and Hardware Requirements

8.1 Software Requirements

- Operating System: Windows 7 or higher, Mac OS
- Java version: JDK 8
- Android Studio
- Android SDK: 1.5 GB or higher
- IDE: Android Studio
- Technologies: Java, XML, Android
- Database: SQLite
- Debugger: Emulator, Android mobile device

8.2 Hardware Requirements

- Disk space: 120GB or more
- Android phone with API level 16 or higher (Jelly bean)

- Processor: i3 or higher
- Disk space for Android Studio: 20 GB
- Space for Android SDK: 1.5 GB or higher
- Processor speed: 2.3 GHz
- RAM: 8GB

9. Methodology Used

Definition: The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. This model is divided into different phases and the output of one phase is used as the input of the next phase. Every phase has to be completed before the next phase starts and there is no overlapping of the phases.

Description: The sequential phases described in the Waterfall model are:

1. *Requirement Gathering*- All possible requirements are captured in product requirement documents.

2. *Analysis Read* - the requirement and based on analysis define the schemas, models and business rules.

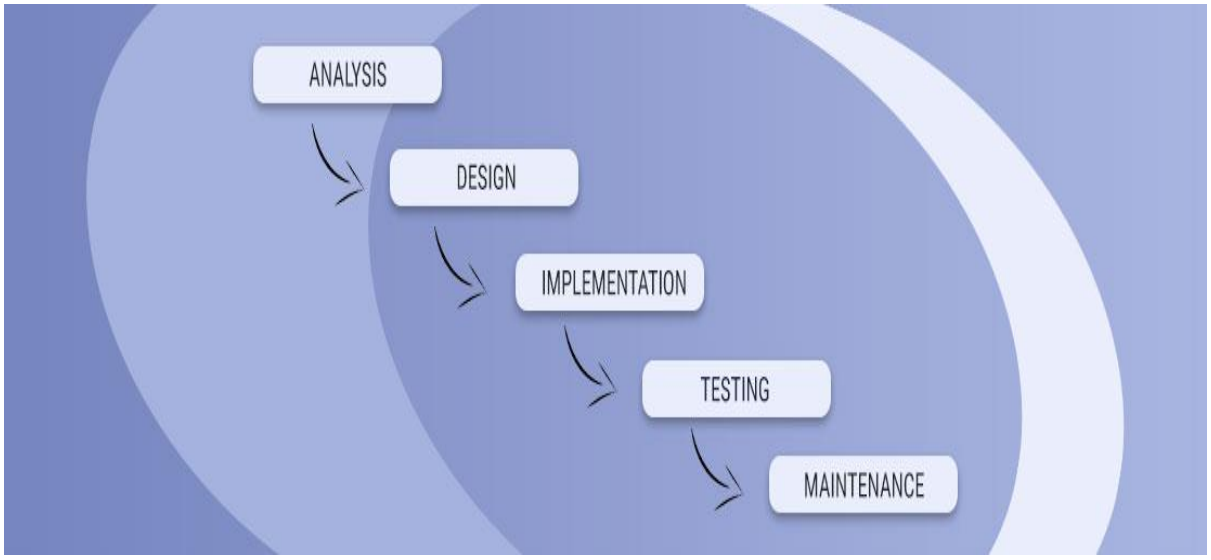
3. *System Design* - Based on analysis design the software architecture.

4. Implementation Development of the software in the small units with functional testing.

5. Integration and Testing Integrating of each unit developed in previous phase and post integration test the entire system for any faults.

6. *Deployment of system* - Make the product live on production environment after all functional and non-functional testing completed.

7. Maintenance Fixing issues and release new version with the issue patches as required.



Waterfall values solid planning. It is a linear and sequential app development methodology in which each project task follows after the previous one is completed.

Waterfall projects are accomplished in a single and very long cycle. Project managers are responsible for thoroughly planning project execution and running the project in a strict sequence based on the requirements specifications. In cases of failed scenarios, the development teams have to repeat all stages, from design to implementation.

All the requirements are defined at the beginning of the project and each phase is completed before moving towards the next stage. Project managers track the progress and make sure there is no overlapping in the stages.

Waterfall Advantages:

Clear structure. When compared with other methodologies, Waterfall focuses mostly on a clear, defined set of steps. Even before the development process starts, the design is hammered out in detail which makes the needs and the outcome clear to everyone in the team.

Easy to manage and control. Each stage has its own reliability and procedures as well as specific deliverable and review processes. Due to the clarity of all the stages, Waterfall is easy to manage. All the tasks can be arranged easily by following the hierarchy.

QA tests. Test scenarios are already defined in the functional specification of the project, which makes the testing process easier and more transparent. If a bug is detected during a certain stage, it is fixed instantly, and the problem gets resolved.

Waterfall Disadvantages:

1. It doesn't allow much reflection or revision. When the product is in testing phase, it is very difficult to go back and change something which is

left during the requirement analysis phase.

2. Risk and uncertainty are high.

3. Not advisable for complex and object-oriented projects.

4. Changing requirements can't be accommodated in any phase.

5. As testing is done at a later phase. So, there is a chance that challenges and risks at earlier phases are not identified.

10. Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

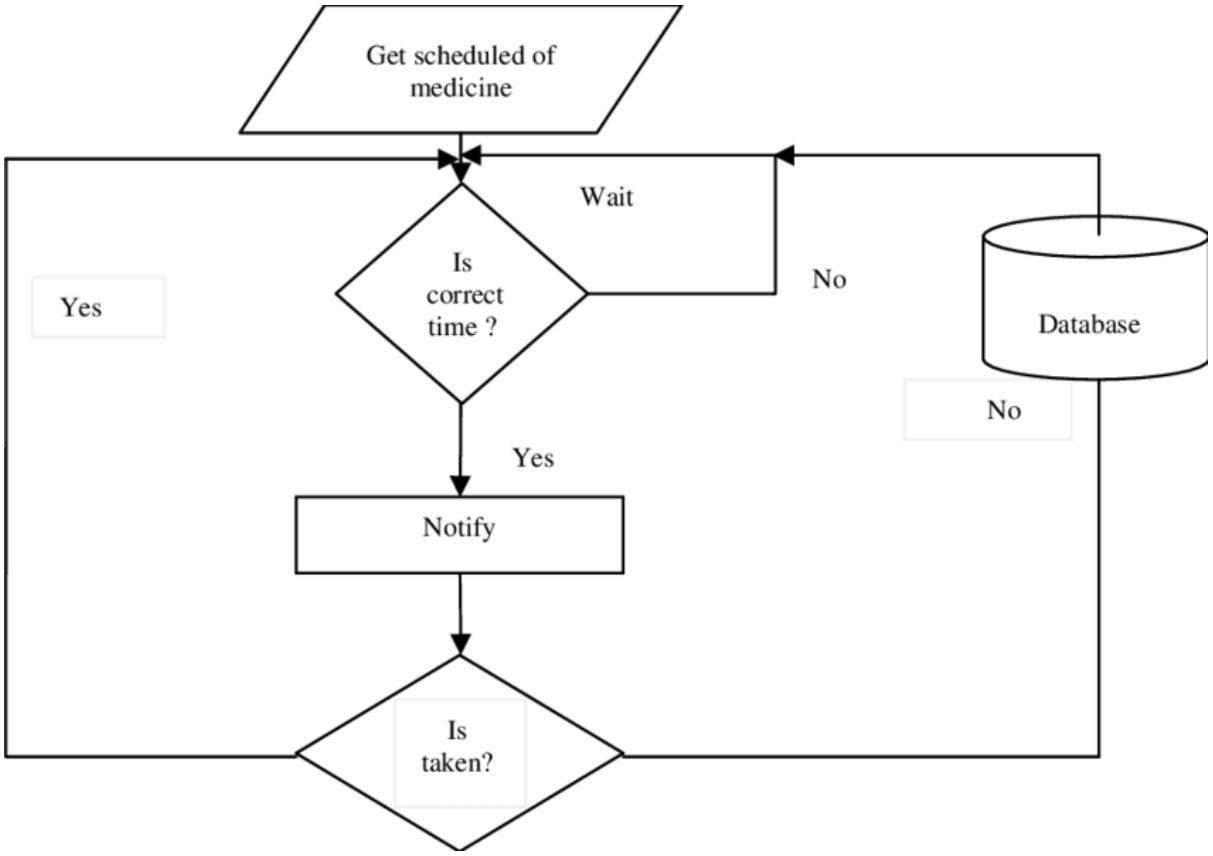
1. **External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

2. **Process:** any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”

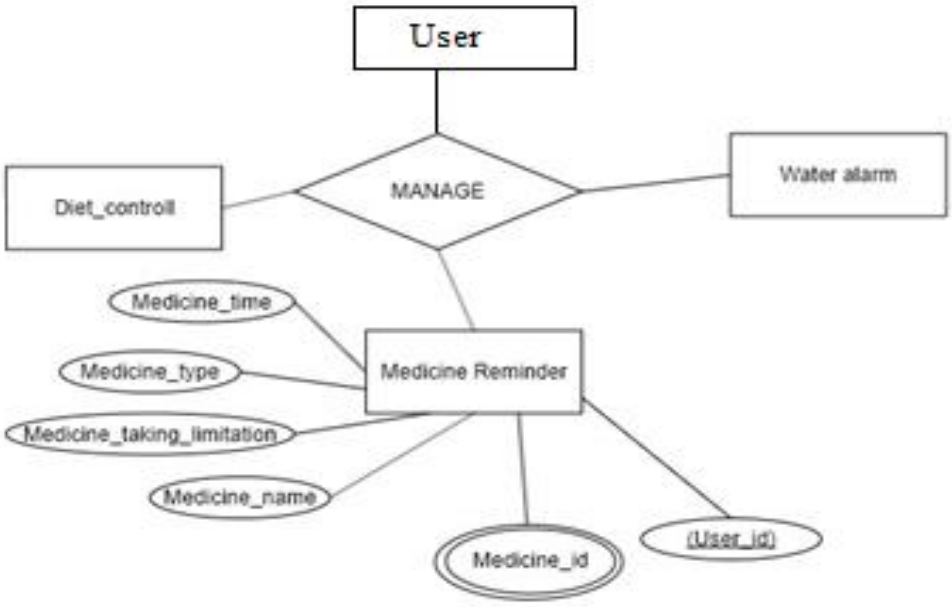
3. **Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”

4. **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

Data-Flow Diagram:



10.1 E-R Diagram



10.2 Requirements Analysis

In this phase, the researcher identified the problems faced with the current system and analyzed, and even predicted potential problems that were likely to arise in future regarding the system. The deliverables / products of this phase drove how the system was to be built and guided the developers' works.

System Design

A. **Overview:** This project aims to develop a healthcare mobile application that provides reminder services to users so that they can adhere to their medication regime. The user creates a new medication schedule by setting a schedule name and specifying the date and time for the first medication notification. Once a schedule is created, additional medication details, such

as medicine name, dosage and audio reminder, can be added into the schedule.

Once the schedule and the medication details have been created, the user will be directed to the view schedule list. While viewing the schedule list, the user can choose to activate the schedule reminder immediately or at a later time. Upon activation of a schedule reminder, the application will be sent to the background. When it is time for the medication alert, the application will play an audio reminder (in the dialect chosen by the user). The application will also display the medication details which include medicine name, dosage and medicine photo in both the notification and main screen of the mobile phone.

B. **Graphical User Interface:** The medication reminder application allows the user to create individual or multiple schedules via a graphical user interface (GUI). In order to create a schedule, the user needs to specify a name for the schedule as well as the date and time to start the alert reminder. Once the schedule is created, the user is directed to a menu where the user can either add or delete medicines from the schedule. To add a medicine, the user simply enters the details of the medicine which include the medicine name, medicine days, dosage and dosage type (tablet/ syrup/etc).If the medicine added is an antibiotic, a checkbox which indicates the course needs to be completed will be checked automatically. The user can select and delete a medicine from the list of medicines that is previously added into the schedule. Deleted medicines are removed from the schedule.

PROPOSED SYSTEM

The idea deals with the development of a system that will help a patient or a person in any medical vulnerability by means of providing them constant reminders of dosage, and providing a reminder about medicine with other health related issues. The intention of this system is to provide a comprehensive understanding of how the Android based medicine Reminder system is built.

Android is a Linux-based operating system designed primarily for touch screen mobile devices such as smart phones and tablet computers, developed by Google in conjunction with the Open Handset Alliance. Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. The system is specified on android operating system only because the market share of Android is high. [9] Android also comes with an application development framework (ADF), which provides an API for application development and includes services for building GUI applications, data access, and other component types. The framework is designed to simplify the reuse and integration of components. Android apps are built using a mandatory XML manifest file. The manifest file values are bound to the application at compile time. This file provides essential information to an Android platform for managing the life cycle of an application. Examples of the kinds of information included in a manifest file are descriptions of the app's components among other architectural and configuration properties. Components can be one of the following types: Activities, Services, Broadcast Receivers, and Content Providers.

Implementation phase

Android is a software platform and Linux-based operating system (OS) designed primarily for touchscreen mobile devices. Its OS is upgradeable and new features are usually available in each new version release. In addition, Android is open-source which is freely available for application developers. The increasing demand of Android devices makes it a great platform to leverage on. This medicine reminder mobile application has been designed to run on the Android platform. It has been deployed and tested successfully on 3 mobile devices: Redmi Note 4, Realme 5 and Galaxy S.

Data Analysis

Data was collected and then resolved conflicts in either requirements of the system and the information provided by the various stakeholders. Questionnaires and interview guide were used to find out how patients have been managed and reminded on their medications

11. Source Code

11.1 XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="GradleMigrationSettings" migrationVersion="1" />
  <component name="GradleSettings">
    <option name="linkedExternalProjectsSettings">
      <GradleProjectSettings>
        <option name="testRunner" value="PLATFORM" />
        <option name="distributionType" value="DEFAULT_WRAPPED" />
        <option name="externalProjectPath" value="$PROJECT_DIR$" />
        <option name="modules">
          <set>
            <option value="$PROJECT_DIR$" />
            <option value="$PROJECT_DIR$/app" />
          </set>
        </option>
        <option name="resolveModulePerSourceSet" value="false" />
      </GradleProjectSettings>
    </option>
  </component>
</project>
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="NullableNotNullManager">
    <option name="myDefaultNullable"
value="android.support.annotation.Nullable" />
    <option name="myDefaultNotNull"
value="androidx.annotation.NonNull" />
    <option name="myNullables">
      <value>
        <list size="12">
          <item index="0" class="java.lang.String"
itemvalue="org.jetbrains.annotations.Nullable" />
          <item index="1" class="java.lang.String"
itemvalue="javax.annotation.Nullable" />
          <item index="2" class="java.lang.String"
itemvalue="edu.umd.cs.findbugs.annotations.Nullable" />
          <item index="3" class="java.lang.String"
itemvalue="android.support.annotation.Nullable" />
          <item index="4" class="java.lang.String"
itemvalue="javax.annotation.CheckForNull" />
          <item index="5" class="java.lang.String"
itemvalue="androidx.annotation.Nullable" />
          <item index="6" class="java.lang.String"
itemvalue="android.annotation.Nullable" />
          <item index="7" class="java.lang.String"
```

```
itemvalue="androidx.annotation.RecentlyNullable" />
          <item index="8" class="java.lang.String"
itemvalue="org.checkerframework.checker.nullness.qual.Nullable" />
          <item index="9" class="java.lang.String"
itemvalue="org.checkerframework.checker.nullness.compatqual.Nullable
Decl" />
          <item index="10" class="java.lang.String"
itemvalue="org.checkerframework.checker.nullness.compatqual.Nullable
Type" />
          <item index="11" class="java.lang.String"
itemvalue="com.android.annotations.Nullable" />
        </list>
      </value>
    </option>

Misc.xml
 <option name="myNotNulls">
      <value>
        <list size="11">
          <item index="0" class="java.lang.String"
itemvalue="org.jetbrains.annotations.NotNull" />
          <item index="1" class="java.lang.String"
itemvalue="javax.annotation.Nonnull" />
          <item index="2" class="java.lang.String"
itemvalue="edu.umd.cs.findbugs.annotations.NonNull" />
          <item index="3" class="java.lang.String"
itemvalue="android.support.annotation.NonNull" />
          <item index="4" class="java.lang.String"
itemvalue="androidx.annotation.NonNull" />
          <item index="5" class="java.lang.String"
itemvalue="android.annotation.NonNull" />
          <item index="6" class="java.lang.String"
itemvalue="androidx.annotation.RecentlyNonNull" />
          <item index="7" class="java.lang.String"
itemvalue="org.checkerframework.checker.nullness.qual.NonNull" />
          <item index="8" class="java.lang.String"
itemvalue="org.checkerframework.checker.nullness.compatqual.NonNullD
ecl" />
          <item index="9" class="java.lang.String"
itemvalue="org.checkerframework.checker.nullness.compatqual.NonNullT
ype" />
          <item index="10" class="java.lang.String"
itemvalue="com.android.annotations.NonNull" />
        </list>
      </value>
    </option>
  </component>
  <component name="ProjectRootManager" version="2"
languageLevel="JDK_1_8" project-jdk-name="1.8" project-jdk-
type="JavaSDK" />
</project>
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="RunConfigurationProducerService">
    <option name="ignoredProducers">
      <set>
```

```
        <option
value="com.android.tools.idea.compose.preview.runconfiguration.Compo
sePreviewRunConfigurationProducer" />
        <option
value="org.jetbrains.plugins.gradle.execution.test.runner.AllInPacka
geGradleConfigurationProducer" />
        <option
value="org.jetbrains.plugins.gradle.execution.test.runner.TestClassG
radleConfigurationProducer" />
        <option
value="org.jetbrains.plugins.gradle.execution.test.runner.TestMethod
GradleConfigurationProducer" />
      </set>
    </option>
  </component>
</project>
```

11.2 Android (Java)

```java
package com.gautam.medicinetime;

import android.app.Application;
import android.content.Context;

/**
 * Created by gautam on 12/07/17.
 */

public class MedicineApp extends Application {

    private static Context mInstance;


    @Override
    public void onCreate() {
        super.onCreate();
        if (mInstance == null) {
            mInstance = getApplicationContext();
        }
    }

    public static Context getInstance() {
        return mInstance;
    }
}
```

```
package com.gautam.medicinetime;
package com.gautam.medicinetime;

/**
 * Created by gautam on 12/07/17.
 */

public interface BasePresenter {

    void start();
}

package com.gautam.medicinetime.medicine;


import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.Nullable;

import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.snackbar.Snackbar;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.gautam.medicinetime.R;
import com.gautam.medicinetime.addmedicine.AddMedicineActivity;
import com.gautam.medicinetime.data.source.MedicineAlarm;
import com.gautam.medicinetime.views.RobotoLightTextView;

import java.util.ArrayList;
import java.util.Calendar;
```

```
import java.util.List;
import java.util.Objects;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import butterknife.Unbinder;

/**
 * Created by gautam on 13/07/17.
 */

public class MedicineFragment extends Fragment implements MedicineContract.Vie
w, MedicineAdapter.OnItemClickListener {

    @BindView(R.id.medicine_list)
    RecyclerView rvMedList;

    Unbinder unbinder;

    @BindView(R.id.noMedIcon)
    ImageView noMedIcon;

    @BindView(R.id.noMedText)
    RobotoLightTextView noMedText;

    @BindView(R.id.add_med_now)
    TextView addMedNow;

    @BindView(R.id.no_med_view)
    View noMedView;

    @BindView(R.id.progressLoader)
    ProgressBar progressLoader;


    private MedicineContract.Presenter presenter;

    private MedicineAdapter medicineAdapter;


    public MedicineFragment() {

    }

    public static MedicineFragment newInstance() {
        Bundle args = new Bundle();
        MedicineFragment fragment = new MedicineFragment();
        fragment.setArguments(args);
```

```
        return fragment;
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        medicineAdapter = new MedicineAdapter(new ArrayList<>(0));
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup cont
ainer, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_medicine, container, fa
lse);
        unbinder = ButterKnife.bind(this, view);
        setAdapter();
        return view;
    }

    @Override
    public void onActivityCreated(@Nullable Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        FloatingActionButton fab = Objects.requireNonNull(getActivity()).findV
iewById(R.id.fab_add_task);
        fab.setImageResource(R.drawable.ic_add);
        fab.setOnClickListener(v -> presenter.addNewMedicine());
    }

    private void setAdapter() {
        rvMedList.setAdapter(medicineAdapter);
        rvMedList.setLayoutManager(new LinearLayoutManager(getContext()));
        rvMedList.setHasFixedSize(true);
        medicineAdapter.setOnItemClickListener(this);
    }

    @Override
    public void onResume() {
        super.onResume();
        Calendar calendar = Calendar.getInstance();
        int day = calendar.get(Calendar.DAY_OF_WEEK);
        presenter.onStart(day);
    }

    @Override
    public void setPresenter(MedicineContract.Presenter presenter) {
        this.presenter = presenter;
    }
```

```
@Override
public void showLoadingIndicator(boolean active) {
    if (getView() == null) {
        return;
    }
    progressLoader.setVisibility(active ? View.VISIBLE : View.GONE);
}

@Override
public void showMedicineList(List<MedicineAlarm> medicineAlarmList) {
    medicineAdapter.replaceData(medicineAlarmList);
    rvMedList.setVisibility(View.VISIBLE);
    noMedView.setVisibility(View.GONE);
}

@Override
public void showAddMedicine() {
    Intent intent = new Intent(getContext(), AddMedicineActivity.class);
    startActivityForResult(intent, AddMedicineActivity.REQUEST_ADD_TASK);
}


@Override
public void showMedicineDetails(long taskId, String medName) {
    Intent intent = new Intent(getContext(), AddMedicineActivity.class);
    intent.putExtra(AddMedicineActivity.EXTRA_TASK_ID, taskId);
    intent.putExtra(AddMedicineActivity.EXTRA_TASK_NAME, medName);
    startActivity(intent);
}


@Override
public void showLoadingMedicineError() {
    showMessage(getString(R.string.loading_tasks_error));
}

@Override
public void showNoMedicine() {
    showNoTasksViews(
            getResources().getString(R.string.no_medicine_added)
    );
}

@Override
public void showSuccessfullySavedMessage() {
    showMessage(getString(R.string.successfully_saved_me_message));
}
```

```
    @Override
    public void showMedicineDeletedSuccessfully() {
        showMessage(getString(R.string.successfully_deleted_message));
        Calendar calendar = Calendar.getInstance();
        int day = calendar.get(Calendar.DAY_OF_WEEK);
        presenter.onStart(day);
    }

    private void showMessage(String message) {
        if (getView() != null)
            Snackbar.make(getView(), message, Snackbar.LENGTH_LONG).show();
    }

    @Override
    public boolean isActive() {
        return isAdded();
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        unbinder.unbind();
    }

    @OnClick(R.id.add_med_now)
    void addMedicine() {
        showAddMedicine();
    }

    private void showNoTasksViews(String mainText) {
        rvMedList.setVisibility(View.GONE);
        noMedView.setVisibility(View.VISIBLE);
        noMedText.setText(mainText);
        noMedIcon.setImageDrawable(getResources().getDrawable(R.drawable.icon_
my_health));
        addMedNow.setVisibility(View.VISIBLE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data)
 {
        presenter.result(requestCode, resultCode);
    }

    @Override
    public void onMedicineDeleteClicked(MedicineAlarm medicineAlarm) {
        presenter.deleteMedicineAlarm(medicineAlarm, getActivity());
    }
```

```
}
```

```
package com.gautam.medicinetime.medicine;

import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.util.Log;

import androidx.annotation.NonNull;

import com.gautam.medicinetime.addmedicine.AddMedicineActivity;
import com.gautam.medicinetime.alarm.ReminderActivity;
import com.gautam.medicinetime.alarm.ReminderFragment;
import com.gautam.medicinetime.data.source.MedicineAlarm;
import com.gautam.medicinetime.data.source.MedicineDataSource;
import com.gautam.medicinetime.data.source.MedicineRepository;

import java.util.Collections;
import java.util.List;
import java.util.Objects;

import static android.content.Context.ALARM_SERVICE;

/**
 * Created by gautam on 13/07/17.
 */

public class MedicinePresenter implements MedicineContract.Presenter {

    private final MedicineRepository mMedicineRepository;

    private final MedicineContract.View mMedView;


    MedicinePresenter(@NonNull MedicineRepository medicineRepository, @NonNull
 MedicineContract.View medView) {
        this.mMedicineRepository = medicineRepository;
        this.mMedView = medView;
        medView.setPresenter(this);
    }

    @Override
```

```
    public void loadMedicinesByDay(int day, boolean showIndicator) {
        loadListByDay(day, showIndicator);
    }

    @Override
    public void deleteMedicineAlarm(MedicineAlarm medicineAlarm, Context conte
xt) {
        List<MedicineAlarm> alarms = mMedicineRepository.getAllAlarms(medicine
Alarm.getPillName());
        for (MedicineAlarm alarm : alarms) {
            mMedicineRepository.deleteAlarm(alarm.getId());
            /** This intent invokes the activity ReminderActivity, which in tu
rn opens the AlertAlarm window */
            Intent intent = new Intent(context, ReminderActivity.class);
            intent.putExtra(ReminderFragment.EXTRA_ID, alarm.getAlarmId());

            PendingIntent operation = PendingIntent.getActivity(context, alarm
.getAlarmId(), intent, PendingIntent.FLAG_UPDATE_CURRENT);


            /** Getting a reference to the System Service ALARM_SERVICE */
            AlarmManager alarmManager = (AlarmManager) Objects.requireNonNull(
context).getSystemService(ALARM_SERVICE);
            if (alarmManager != null) {
                alarmManager.cancel(operation);
            }
        }
        mMedView.showMedicineDeletedSuccessfully();
    }

    @Override
    public void start() {

    }

    @Override
    public void onStart(int day) {
        Log.d("TAG", "onStart: " + day);
        loadMedicinesByDay(day, true);
    }

    @Override
    public void reload(int day) {
        Log.d("TAG", "reload: " + day);
        loadListByDay(day, true);
    }

    @Override
    public void result(int requestCode, int resultCode) {
```

```
        if (AddMedicineActivity.REQUEST_ADD_TASK == requestCode && Activity.RE
SULT_OK == resultCode) {
            mMedView.showSuccessfullySavedMessage();
        }
    }

    @Override
    public void addNewMedicine() {
        mMedView.showAddMedicine();
    }

    private void loadListByDay(int day, final boolean showLoadingUi) {
        if (showLoadingUi)
            mMedView.showLoadingIndicator(true);

        mMedicineRepository.getMedicineListByDay(day, new MedicineDataSource.L
oadMedicineCallbacks() {
            @Override
            public void onMedicineLoaded(List<MedicineAlarm> medicineAlarmList
) {
                processMedicineList(medicineAlarmList);
                // The view may not be able to handle UI updates anymore
                if (!mMedView.isActive()) {
                    return;
                }
                if (showLoadingUi) {
                    mMedView.showLoadingIndicator(false);
                }
            }

            @Override
            public void onDataNotAvailable() {
                if (!mMedView.isActive()) {
                    return;
                }
                if (showLoadingUi) {
                    mMedView.showLoadingIndicator(false);
                }

                mMedView.showNoMedicine();
            }
        });
    }

    private void processMedicineList(List<MedicineAlarm> medicineAlarmList) {

        if (medicineAlarmList.isEmpty()) {
```

```
            // Show a message indicating there are no tasks for that filter ty
pe.
            mMedView.showNoMedicine();
        } else {
            //Show the list of Medicines
            Collections.sort(medicineAlarmList);
            mMedView.showMedicineList(medicineAlarmList);
        }
    }

}
```

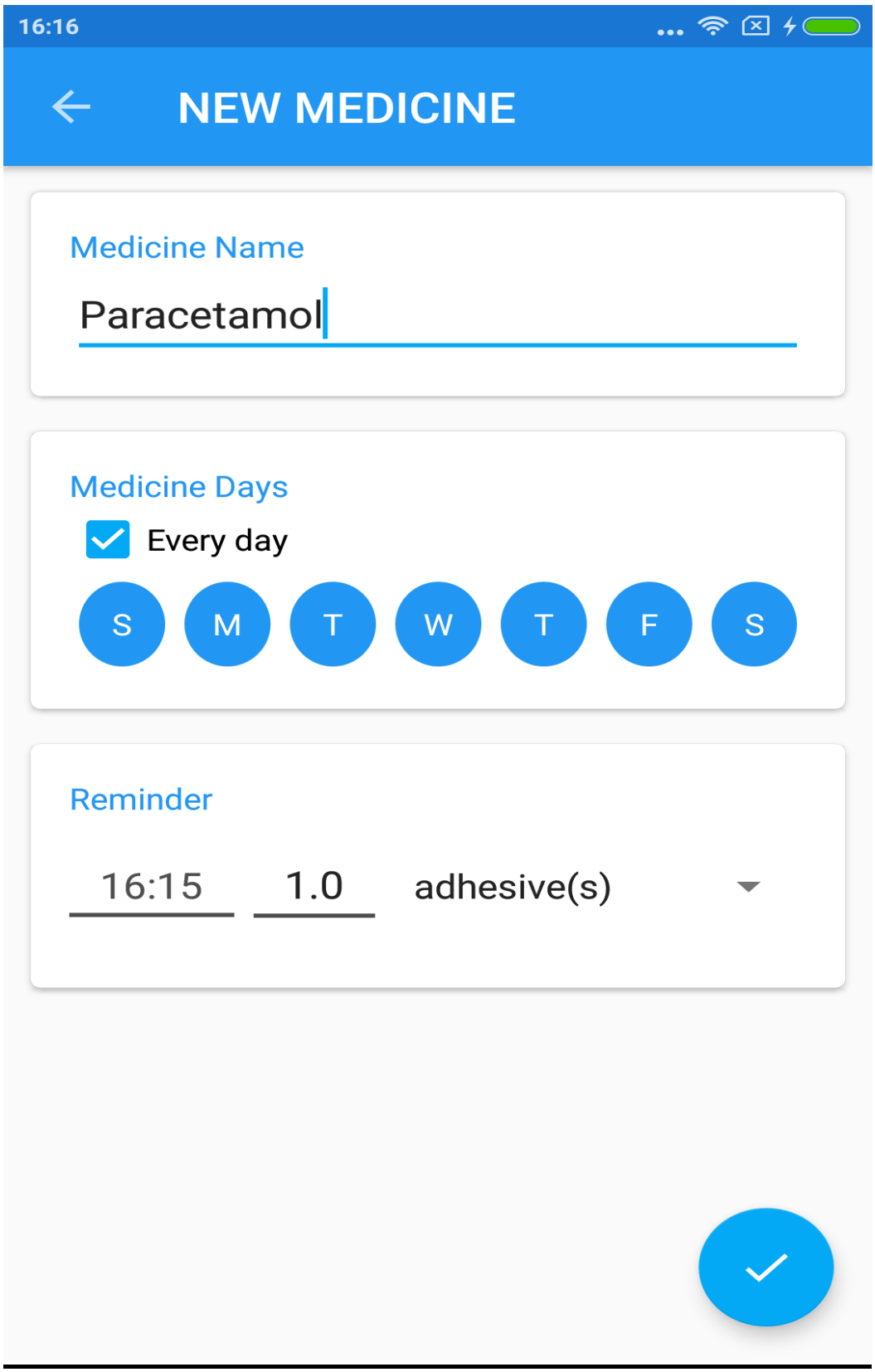
```
package com.gautam.medicinetime.medicine;

import android.content.Intent;
import android.os.Bundle;
import com.google.android.material.appbar.AppBarLayout;
import com.google.android.material.appbar.CollapsingToolbarLayout;
import androidx.coordinatorlayout.widget.CoordinatorLayout;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import androidx.core.view.ViewCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.gautam.medicinetime.Injection;
import com.gautam.medicinetime.R;
import com.gautam.medicinetime.report.MonthlyReportActivity;
import com.gautam.medicinetime.utils.ActivityUtils;
import com.github.sundeepk.compactcalendarview.CompactCalendarView;

import java.text.SimpleDateFormat;
import java.util.Calendar;
```

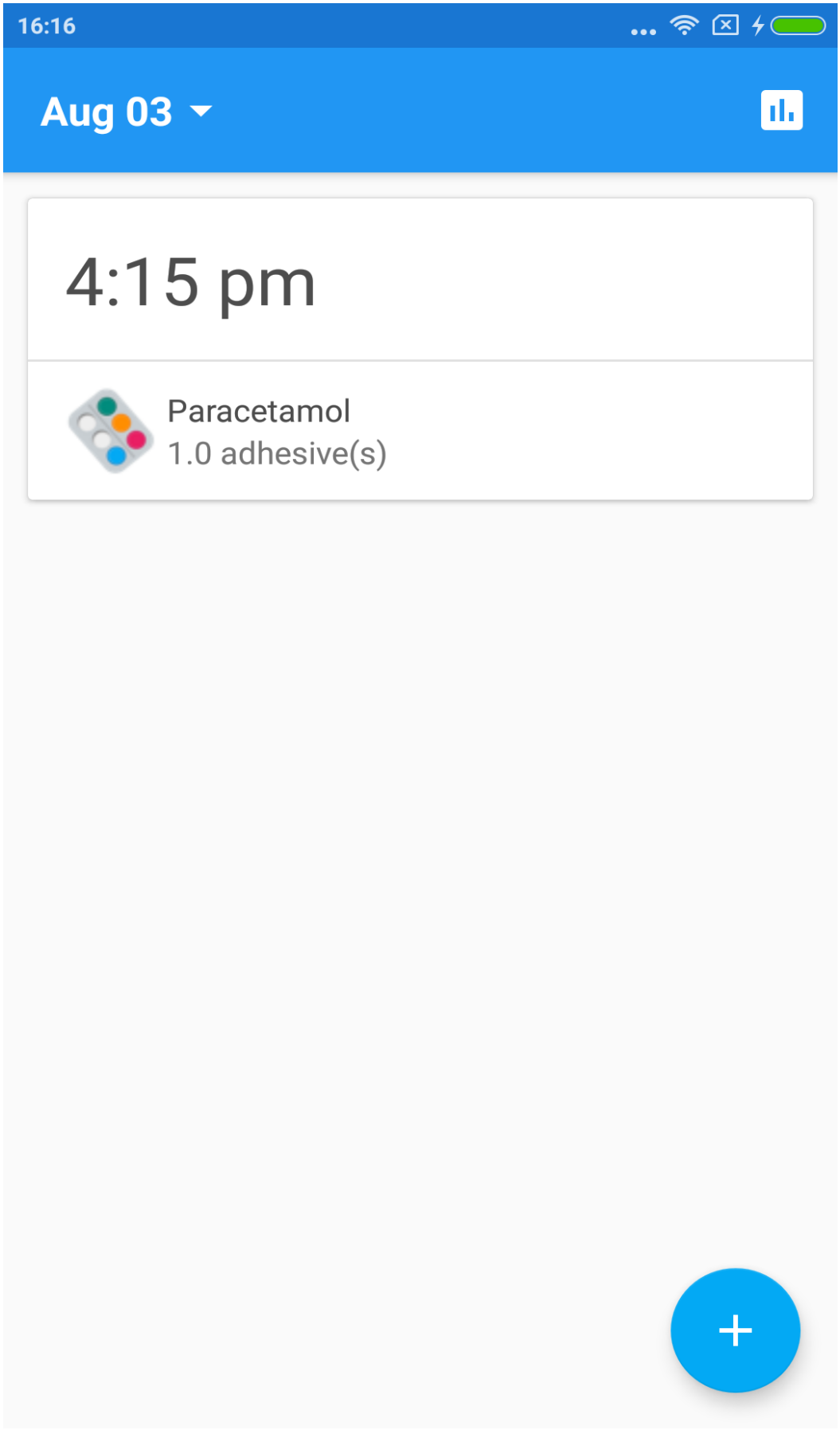
```
import java.util.Date;
import java.util.Locale;
import java.util.TimeZone;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class MedicineActivity extends AppCompatActivity {


    @BindView(R.id.compactcalendar_view)
    CompactCalendarView mCompactCalendarView;

    @BindView(R.id.date_picker_text_view)
    TextView datePickerTextView;

    @BindView(R.id.date_picker_button)
    RelativeLayout datePickerButton;

    @BindView(R.id.toolbar)
    Toolbar toolbar;

    @BindView(R.id.collapsingToolbarLayout)
    CollapsingToolbarLayout collapsingToolbarLayout;

    @BindView(R.id.app_bar_layout)
    AppBarLayout appBarLayout;

    @BindView(R.id.contentFrame)
    FrameLayout contentFrame;

    @BindView(R.id.fab_add_task)
    FloatingActionButton fabAddTask;

    @BindView(R.id.coordinatorLayout)
    CoordinatorLayout coordinatorLayout;

    @BindView(R.id.date_picker_arrow)
    ImageView arrow;

    private MedicinePresenter presenter;

    private SimpleDateFormat dateFormat = new SimpleDateFormat("MMM dd", /*Loc
ale.getDefault()*/Locale.ENGLISH);

    private boolean isExpanded = false;

    @Override
```

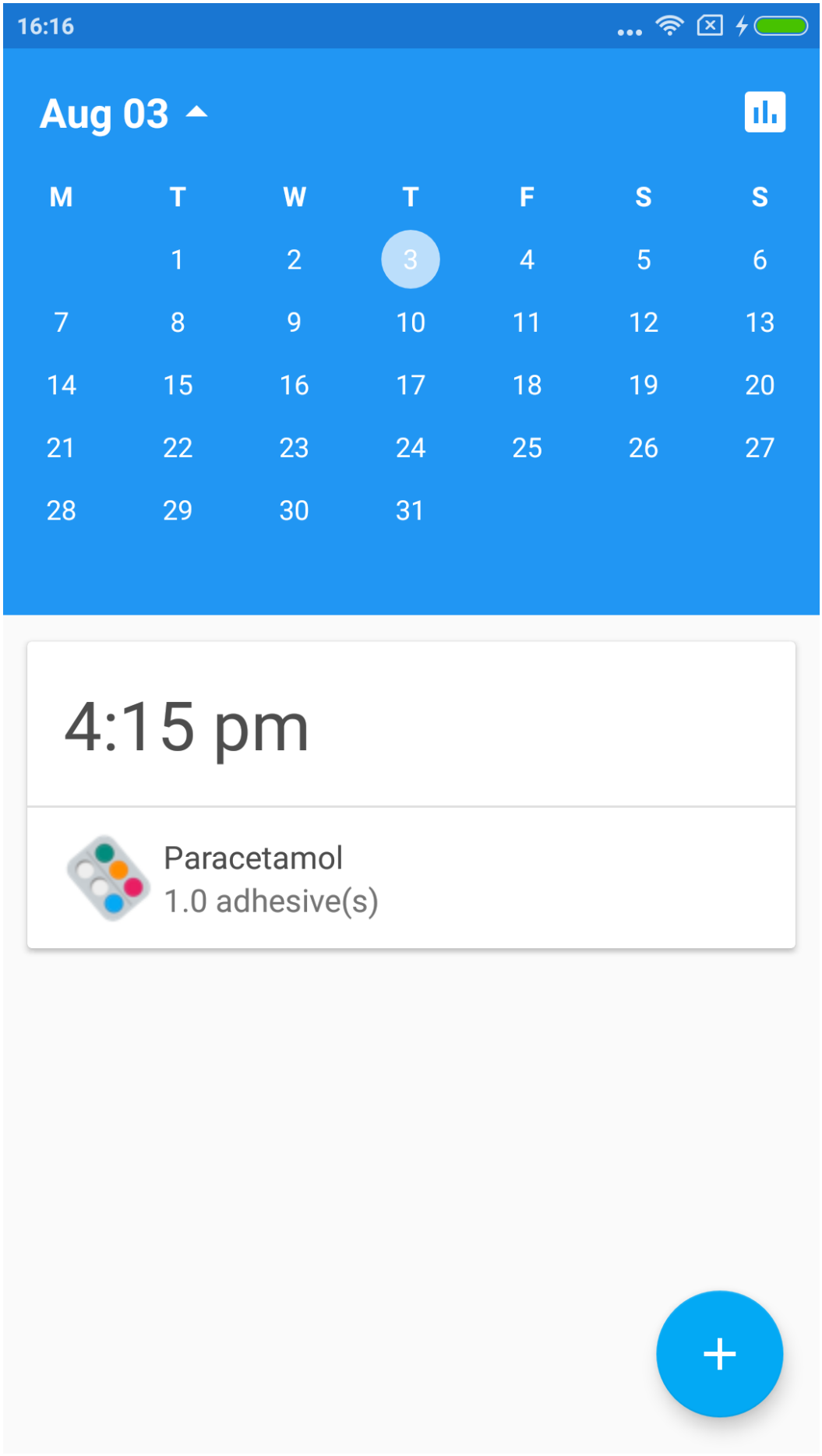
```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_medicine);
        ButterKnife.bind(this);
        setSupportActionBar(toolbar);

        mCompactCalendarView.setLocale(TimeZone.getDefault(), /*Locale.getDefa
ult()*/Locale.ENGLISH);

        mCompactCalendarView.setShouldDrawDaysHeader(true);

        mCompactCalendarView.setListener(new CompactCalendarView.CompactCalend
arViewListener() {
            @Override
            public void onDayClick(Date dateClicked) {
                setSubtitle(dateFormat.format(dateClicked));
                Calendar calendar = Calendar.getInstance();
                calendar.setTime(dateClicked);

                int day = calendar.get(Calendar.DAY_OF_WEEK);

                if (isExpanded) {
                    ViewCompat.animate(arrow).rotation(0).start();
                } else {
                    ViewCompat.animate(arrow).rotation(180).start();
                }
                isExpanded = !isExpanded;
                appBarLayout.setExpanded(isExpanded, true);
                presenter.reload(day);
            }

            @Override
            public void onMonthScroll(Date firstDayOfNewMonth) {
                setSubtitle(dateFormat.format(firstDayOfNewMonth));
            }
        });
        setCurrentDate(new Date());
        MedicineFragment medicineFragment = (MedicineFragment) getSupportFragm
entManager().findFragmentById(R.id.contentFrame);
        if (medicineFragment == null) {
            medicineFragment = MedicineFragment.newInstance();
            ActivityUtils.addFragmentToActivity(getSupportFragmentManager(), m
edicineFragment, R.id.contentFrame);
        }

        //Create MedicinePresenter
        presenter = new MedicinePresenter(Injection.provideMedicineRepository(
MedicineActivity.this), medicineFragment);
```

```
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.medicine_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.menu_stats) {
            Intent intent = new Intent(this, MonthlyReportActivity.class);
            startActivity(intent);
        }
        return super.onOptionsItemSelected(item);
    }

    public void setCurrentDate(Date date) {
        setSubtitle(dateFormat.format(date));
        mCompactCalendarView.setCurrentDate(date);
    }

    public void setSubtitle(String subtitle) {
        datePickerTextView.setText(subtitle);
    }

    @OnClick(R.id.date_picker_button)
    void onDatePickerButtonClicked() {
        if (isExpanded) {
            ViewCompat.animate(arrow).rotation(0).start();
        } else {
            ViewCompat.animate(arrow).rotation(180).start();
        }

        isExpanded = !isExpanded;
        appBarLayout.setExpanded(isExpanded, true);
    }
}
```

12. Screenshots




16:16
Aug 03
4:15 pm
Paracetamol
1.0 adhesive(s)


16:16
Aug 03
M T W T F S S
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
4:15 pm
Paracetamol
1.0 adhesive(s)

13. Result and Discussion

The application gives reliable reminders, good user interface, nice user experience and it supports many new features supporting medication adherence. This showed that the combination of all the functionalities provided is useful to the people of all ages.

The users will get the schedule of medicine in-take time with medicine description, starting and ending date of medicine, automatic alarm ringing system and navigation system. So this was found beneficial to the people of any age group. The automatic alarm ringing feature was proved beneficial to 100% of the total population. The youths are very much concerned with the new health care awareness and are interested in knowing about new medical techniques being developed every day.

The medicine reminder system serves reliable reminders, has a good and easy to use user interface and supports a lot of features adhering to medicines. The details are not at all confusing and can be easily understood by the user.

Discussion

While working on this project, we faced several challenges. First of all gathering medical data was the worst hustle. That is why we had to take most of the data from the internet. Although we collected all the data from the best possible source, it would have been better if we could use real data. We have tried our best to overcome all the short comes yet there is always a huge scope of improvement.

The Medicine Reminder mobile android app is implemented so that users are involved in the improvement of medication adherence. Certain challenges like

security and privacy issues have been taken care of in this app as all the data is stored in the internal storage and is only accessible by the users and the app itself. This app can be enhanced with potential extensions such as connecting the app with the cloud and generalizing the app in such a way that the user can add more medications. Connecting this app with the cloud would help users track the medication intake more frequently resulting in enhanced health care.

14. Advantages

Cost Efficient: Our product cost is affordable compare to other product available in market. User friendly: User can set time table of medicine by himself.

Highly Reliable: Good in quality and performance; able to be trusted for patients & old age people.

Provide Comfort and Health: Comfortable for old age people and provide healthy life for patients who are regularly take medicines.

Long-Lasting: The product can be used for long time.

Accurate Result: Alarm will ring at proper time which is set by user previously. Easy to maintain: It need less Maintenance. It is one time investment afterwards it can be used continuously.

15. Future Scope and Further Enhancement

In future our dream is to make this application more useful, effortless, attractive and providing more features.

We are looking forward to add social media connectivity to the application so that using the app becomes more fun and enjoyable. We are also trying to build a direct easy communication between patients and the hospital or other health service providers for better and easier health service.

We also have plans to integrate e-commerce business with our system such as online pharmacy, so that the project can make money and we can continue its development.

We have developed the best health app in this region with so many services. We have many ideas that will take time to complete because research is needed. We want it to be fully automated. We want to do research on it because this project aims to serve the people. We want to add the following features in our application.

1.We want to make a native react.js version of our app which will run in all platforms.

2.We want to add machine learning and AI into my app so that users will enjoy many services automatically.

16. Conclusion

The usefulness of this mobile application depends on usability and user acceptance. Although the app is still in development phase, we have tried our best to make as easy to use as possible. We are continuously trying to make the application better and better day by day. We are eagerly waiting for that day when our app will be available in the app stores and we see that people using the application and getting profit from it.

The medication reminder application can impact positively on the life of the patient as it will help the patient in keeping track of their daily intake of pills as remembering the intake of the prescribed medication can be a matter of life and death. While the app provides an array of options to customizing the booze ringing tone, turn off the notifications and could see missed dose which is absent in some application designs.

Medicine Reminder application is hoped to help patients to incorporate their medication management responsibility into their smartphone-friendly lifestyle through its dual functionality as reminder for medication as well medicine details add. Medicine Reminder app is therefore hoped to improve the overall health quality in the long run.

17. References

- Zao J.K., Mei-Ying Wang, Peihsuan Tsai, Liu J.W.S., *"Smart Phone Based Medicine In-take Scheduler, Reminder and Monitor"*, IEEE e-Health Networking Applications and Services (Healthcom), 2010[1]

- https://play.google.com/store/apps/details?id=com.dambara.medremindpro&feature=search_result[2]

- https://play.google.com/store/apps/details?id=com.garland.medminderfree&feature=search_result[3]

- http://en.wikipedia.org/wiki/Android_%28operating_system%29[4]

- http://www.devworx.in/news/android-news/android-defeats-all-others-in-south-east-asia-market-survey-122301.html[5]

- J. M. Slagle, J. S. Gordon, C, E, Harris, C. L. Davison, D. K. Culpepper, P. Scott P and K. B. Johnson [6], MyMediHealth – Designing a next generation system for child-centered medication management, Journal of Biomedical Informatics, vol. 43, no. 5, **(2011)**, pp. 27 31.
- Alexander Backlund, (2000)[7], the definition of system. In: *Kybernetes* Vol. 29 nr. 4, pp. 444–451.
- Dayer, L., Heldenbrand, S., Anderson, P., Gubbins, P. O., and Martin, B. C. Smart- phone medication adherence apps: Potential benefits to patients

and providers. Journal of the American Pharmacists Association, 53, 2 (2013), 172–181.

- Laird, S. (2012). How smartphones are changing healthcare. Retrieved 11/07/2013, from http://mashable.com/2012/09/26/smartphones-health-careinfographic/
- Obiodu, V., & Obiodu, E. An Empirical Review of the Top 500 Medical Apps in a European Android Market. Journal of Mobile Technology in Medicine, 1, 4 (2012), 22–37.
- Silva, J. M., Mouttham, A., and El Saddik, A. UbiMeds: a mobile application to improve accessibility and support medication adherence. Proc. MSIADU'09. ACM Press (2009), 71 78.
- Rosenberg, M. J., & Waugh, M. S. Causes and consequences of oral contraceptive non-compliance. *American Journal of Obstetrics and Gynecology*, 180, 2 Pt 2 (1999), 276–279.