

**PILLS NOTIFY**  
**Main Project Report**

Submitted by  
**Harsha S Pillai**

**Reg No : FIT20MCA-2056**

*Submitted in partial fulfillment of the requirements for the award of  
the degree of*

*Master of Computer Applications  
Of  
A P J Abdul Kalam Technological University*



**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®**  
**ANGAMALY-683577, ERNAKULAM(DIST)**  
**JULY 2022**

## **DECLARATION**

I, **Harsha S Pillai** hereby declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (**FISAT**), Angamaly in partial fulfillment of the award of the degree of Master of Computer Application is an authentic record of my original work.

The report has not been submitted for the award of any degree of this university or any other university.

**Date :**

**Signature :**

**Place : Angamaly**

**Name :**

**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®  
ANGAMALY, ERNAKULAM-683577**

**DEPARTMENT OF COMPUTER APPLICATIONS**



**CERTIFICATE**

This is to certify that the project report titled **"PILLS NOTIFY"** submitted by **Harsha S Pillai [Reg No: FIT20MCA-2056]** towards partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of bonafide work carried out by her during the year 2022.

**Project Guide**

**Head of the Department**

Submitted for the viva-voice held on ..... at .....

---

## ACKNOWLEDGEMENT

I am deeply grateful to **Dr. Manoj George** , Principal, FISAT, Angamaly for providing me with adequate facilities, way and means by which I was able to complete our main project work and I express my sincere thanks to **Dr. C Sheela** ,Vice Principal FISAT, Angamaly.

My sincere thanks to **Dr. Deepa Mary Mathews**, Head of the department of MCA, FISAT, who had been a source of inspiration. I express heartiest thanks to **Ms. Manju Joy** my project guide for the encouragement and valuable suggestion . I express my heartiest gratitude to my scrum master **Dr. Sujesh P Lal** and the faculty members in our department for their constant encouragement and never ending support throughout the project.I would also like to express our sincere gratitude to the lab faculty members for their guidance.

Finally I express my thanks to all my friends who gave me wealth of suggestions for successful completion of this project.

---

## ABSTRACT

Pills notify is an Android-based application in which an automatic alarm ringing system is implemented. 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. A notification will be sent to them at the time of medication. It is possible to add multiple users in a single application.

This system focuses on easy navigation and good user interface. Many such Medical Reminder Systems have been developed where a new hardware is required but here an attempt to develop a system which is economical, time-saving and supports medication adherence was made. In Modern healthcare most of the errors have been identified in Out-patient medication administration. These medication errors are caused due to under or over doses and forgot to take medicines at proper time. Because of these types of errors recovery from the diseases are getting delayed and the patient is suffering for more time. This application will remind the user to take proper medicines in proper quantity at proper time.

Alarms have one time of day and can occur on multiple days of the week. The user is able to view their pills in a today view and can select date to view medicines. Health is the only thing that we should care the most. COVID-19 made all the people aware about the importance of health. In such a situation, forgetting about the medicine timings can severely affect the health of an individual.

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>8</b>
<b>2</b>	<b>PROOF OF CONCEPT</b>	<b>10</b>
<b>3</b>	<b>IMPLEMENTATION</b>	<b>12</b>
3.1	Objective . . . . .	13
3.2	Existing and proposed system . . . . .	14
3.3	Proposed system and Implementation . . . . .	14
3.4	Requirement Analysis Specification . . . . .	17
3.4.1	Hardware Requirements . . . . .	18
3.4.2	Introduction to language . . . . .	19
3.4.3	XML . . . . .	20
3.4.4	Java . . . . .	21
<b>4</b>	<b>TESTING</b>	<b>23</b>
4.1	Unit testing . . . . .	23
4.2	Integration testing . . . . .	27
4.3	System testing . . . . .	29
<b>5</b>	<b>MODULES</b>	<b>32</b>
<b>6</b>	<b>RESULT ANALYSIS</b>	<b>34</b>
<b>7</b>	<b>CONCLUSION</b>	<b>35</b>

---

<b>8</b>	<b>APPENDIX</b>	<b>37</b>
8.1	CODING . . . . .	37
8.1.1	AddNewMedBusinessLayer . . . . .	37
8.1.2	HamburgerBusinessLayer . . . . .	40
8.1.3	UserDetailsBusinessLayer . . . . .	41
8.1.4	ResetApplicationLayer . . . . .	43
8.1.5	DisplayNotification . . . . .	44
8.1.6	SplashScreen . . . . .	47
8.1.7	NotificationDisplayMedicine . . . . .	48
8.1.8	MainActivity . . . . .	50
8.2	SCREENSHOTS . . . . .	52
<b>9</b>	<b>REFERENCES</b>	<b>59</b>

# Chapter 1

## INTRODUCTION

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.

In our developing and technology dependent life we totally rely on gadgets especially smart phones. Today everyone has a smart phone. With this we get an opportunity to use technology in a better way so that it can be made useful to us. And it plays an important part in our daily life and helps us staying fit in many ways.

The remarkable problem is that patients forget to take the proper medicines in proper proportion and in proper time. Medication adherence, which refers to the degree or extent to which a patient takes the right medication at the right time according to a doctor's prescription, has recently emerged as a serious issue because many studies have reported that non-adherence may critically affect the patient, thereby raising medical costs. Medication non adherence is a com-



mon, complex, and costly problem that contributes to poor treatment outcomes and consumes health care resources.

So here an Android application is introducing whose objective is to remind the patients of their dosage timings through Alarm Ringing system so that they can stay fit and healthy. This application focuses on the people who forget to take medicines on time. It allows users to set an alarm along with the fields of date, time and medicine description which will allow them to set alarm for multiple medicines at different time intervals. The notification system will send a notification after setting an alarm. Medication reminders help in decreasing medication dispensing errors and wrong dosages.

## **Chapter 2**

# **PROOF OF CONCEPT**

The use of mobile devices such as handsets, Personal Digital Assistants (PDAs) and tablet computers has increased in the past few years that it has become a must have, in fact mobile phones and other devices has become human companion almost impossible to live without. All professions make use of mobile device in one form or the other and the inclusion of web services within the device has further encouraged its usage. Thus, developers are regularly developing applications that will run on mobile phones. These applications are designed and programmed to meet specific needs and also to run on specific platform. Mobile phone platforms are important as it predetermines the type of applications and software that will eventually run on it. Phone manufacturers are creating phones that supports two platforms and convergence is something that is been advocated for within the industry.

Pills Notify is an Android application which helps to remind users of their daily (weekly, monthly) intake of medicines. This application is mostly targeted at an elderly audience with a focus on accessibility and usability. The user will be able to enter their medical information and prescriptions and the application will send a push notification when it is time for the user to take their inputted medication. I have kept in mind the common isolation of senior citizens while creating Pills Notify. The purpose of the application is to serve as

a medication administration reminder. The primary audience is people aged 55+ and the secondary audience is anyone who has many medications to keep track of.

The application has been designed in such a way that it would be easy to use since the majority of the users will be older people. Thus, this application will act as an assistant that helps the users to remember medication times. The users will also be able to enter personal and prescription details in the application. The users will be reminded of their medications through sound, vibration and even flashlight. The main benefit of using this application is that the users will not need to track times for taking regularly scheduled medicines.

## **Chapter 3**

# **IMPLEMENTATION**

Pills Notify is an android application meant to aid the forgetful and busy with remembering to take their daily medications. It is designed for users who need a little help keeping track of their medication schedule and who are dedicated to keeping the schedule. The application allows the user to store pill objects and multiple alarms for those pills. The user can view their pills in a today view and can select date to view medicines, the application stores the history of when each medication was taken. This will aid the user in keeping track of their medication usage.

It reminds you whenever you need to take a certain pill. It is especially aimed for those, who have trouble remembering the same or are too busy with their own schedule that they do not keep track of time.

## **3.1 Objective**

To remind users of their daily intake of medicines and is mostly targeted at an elderly audience with a focus on accessibility and usability and to serve as a medication administration reminder which act as a scheduler and a reminder app at the same time and the users will not need to track times for taking regularly scheduled medicines.

### **Environment**

The Pills Notify application can be efficiently used in an environment which is calm and noise-free. The noise free environment would allow the users of the application to hear all of the medicine notifications of the application. The interface will be calm and free of clutter to allow the users of the applications to easily see the medicines on the screen. The main environment where the application can be used are homes, hospitals, and at any place which is calm and noise-free.

### **Mode**

The application has been designed for users which are seated on chair or sofa, relaxing at their home, or on-the-go. As Long as the user has their medication on them, which they should, they will be able to use the application effectively. If the user's state of mind is busy, bored or angry, then such users will not enjoy the application. This application is expected to be used when the user receives a notification about their medication, when the user wants to enter medication information, or when they would like to see when they need to administer their next medication.

## **3.2 Existing and proposed system**

The existing system allows the user to store pill objects and alarms for those pills. The user can view their pills in a today view, a tomorrow view, a weekly view, and an editable view that shows every pill and every alarm. It allows the user to add an alarm, edit the alarm and delete the alarm.

The proposed system is more efficient as it allows to store the quantities of medicine so that one can know, how many tablets or medicines he/she must take. The app would use the different ringtones to notify the user about the medication. The app is also implemented with a splash light feature, so that the user will notified at the time of his/her medications.

## **3.3 Proposed system and Implementation**

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. The proposed system will be developed for Android mobiles only because the market share of Android is more than other operating systems.

### Smartphone Share

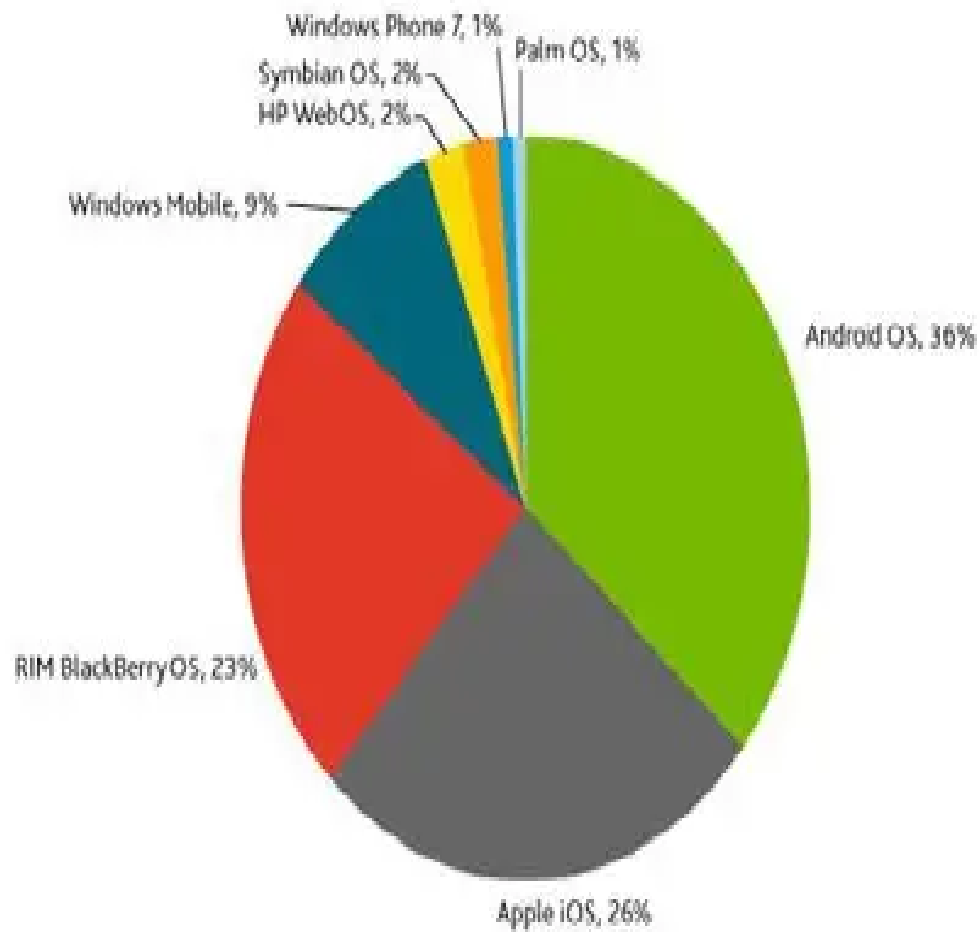


Figure 3.1: Mobile OS Usage Analysis



### **3.4 Requirement Analysis Specification**

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently. The primary phase in the study of any system begins with a stated user need followed by initial investigation of the existing system. As a part of this exercise, meeting with project team, review documents, forms etc. were made. Following up these activities a summarized preliminary report was prepared. Information collected during this phase was used to define the problem, its scope, objective and device a more efficient system.

In this stage of the software development process, potential requirements of the application were identified and defined in terms what the application should do and not how it works. The following are some of the functional requirements of the android based medication reminder and adherence application:

- i. It should notify the users about the incoming messages
- ii. It should allow to add a new user
- iii. It has also the splash light feature along with the notification
- iv. It should allow a user to set a reminder and a notification message.
- v. It should allow to store the medicine details such as name, dosage etc.

## Software Requirement Specifications

- 1) Microsoft windows 8,8.1,10 or 11
- 2) Java Development Kit(JDK)
- 3) The android SDK
- 3) Gradle (an advanced build toolkit that manages dependencies and allows to define custom build logic) NetBeans.

An integrated development environment (IDE) is a software suite that consolidates basic tools required to write and test software. Developers use numerous tools throughout software code creation, building and testing. Development tools often include text editors, code libraries, compilers and test platforms. Without an IDE, a developer must select, deploy, integrate and manage all of these tools separately. An IDE brings many of those development-related tools together as a single framework, application or service.

The integrated toolset is designed to simplify software development and can identify and minimize coding mistakes and typos. Some IDEs are open source, while others are commercial offerings. An IDE can be a standalone application or it can be part of a larger package. We are using ANDROID STUDIO which is the official IDE used for developing android applications.

### 3.4.1 Hardware Requirements

- 1) Dual core 64-bit processor Minimum 4 GB of RAM.
- 2) 1.5 GB of Hard Disk space
- 3) Android phone.

### **3.4.2 Introduction to language**

#### **Android**

Android is a Linux based operating system. It is designed primarily for touch screen mobile devices such as smart phones and tablet computers. Android is a most powerful Operating System supporting a large number of applications. Android is the most developing Operating System in these recent days. Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo are the different versions of android.

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013, at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as go and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects back port some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8

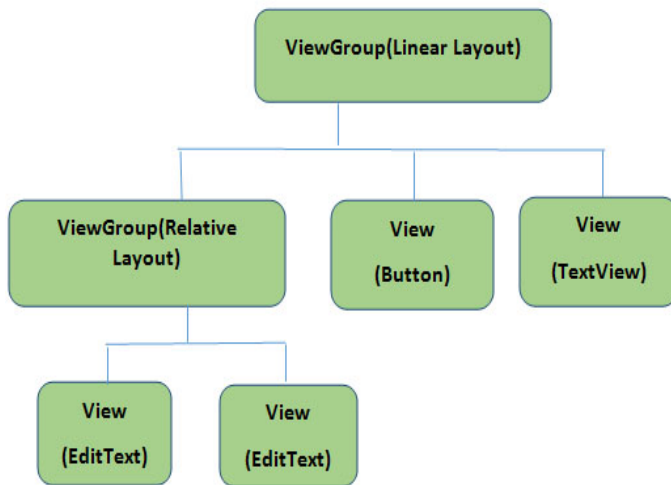
support). At least some new language features up to Java 12 are usable in Android.

Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.

### **3.4.3 XML**

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. It is derived from Standard Generalized Markup Language(SMGL). Basically, the XML tags are not predefined in XML. We need to implement and define the tags in XML. XML tags define the data and used to store and organize data. It's easily scalable and simple to develop. In Android, the XML is used to implement UI-related data, and it's a lightweight markup language that doesn't make layout heavy. XML only contains tags, while implementing they need to be just invoked.

Basically in Android XML is used to implement the UI-related data. So understanding the core part of the UI interface with respect to XML is important. The User Interface for an Android App is built as the hierarchy of main layouts, widgets. The layouts are View Group objects or containers that control how the child view should be positioned on the screen. Widgets here are view objects, such as Buttons and text boxes. The whole concept of Android User Interface is defined using the hierarchy of View and View Group objects. A View Group is an invisible container that organizes child views. These child views are other widgets which are used to make the different parts of UI. One View Group can have another View Group as an child element as shown in the figure given below:



### 3.4.4 Java

Android App are mostly developed in JAVA language using Android SDK (Software Development Kit). Other languages like C, C++, Scala etc. can also be used for developing Android App, but JAVA is most preferred and mostly used programming language for Android App Development. So if you are a beginner in Android then JAVA language and complete knowledge of OOPS concepts is the first thing you need to learn before beginning Android Development.

JAVA is a programming language which is used in Android App Development. It is class based and object oriented programming whose syntax is influenced by C++. The primary goals of JAVA is to be simple, object-oriented, robust, secure and high level.

JAVA application runs on JVM (JAVA Virtual Machine) but Android has it's own virtual machine called Dalvik Virtual Machine (DVM) optimized for mobile devices.

#### DATA FLOW

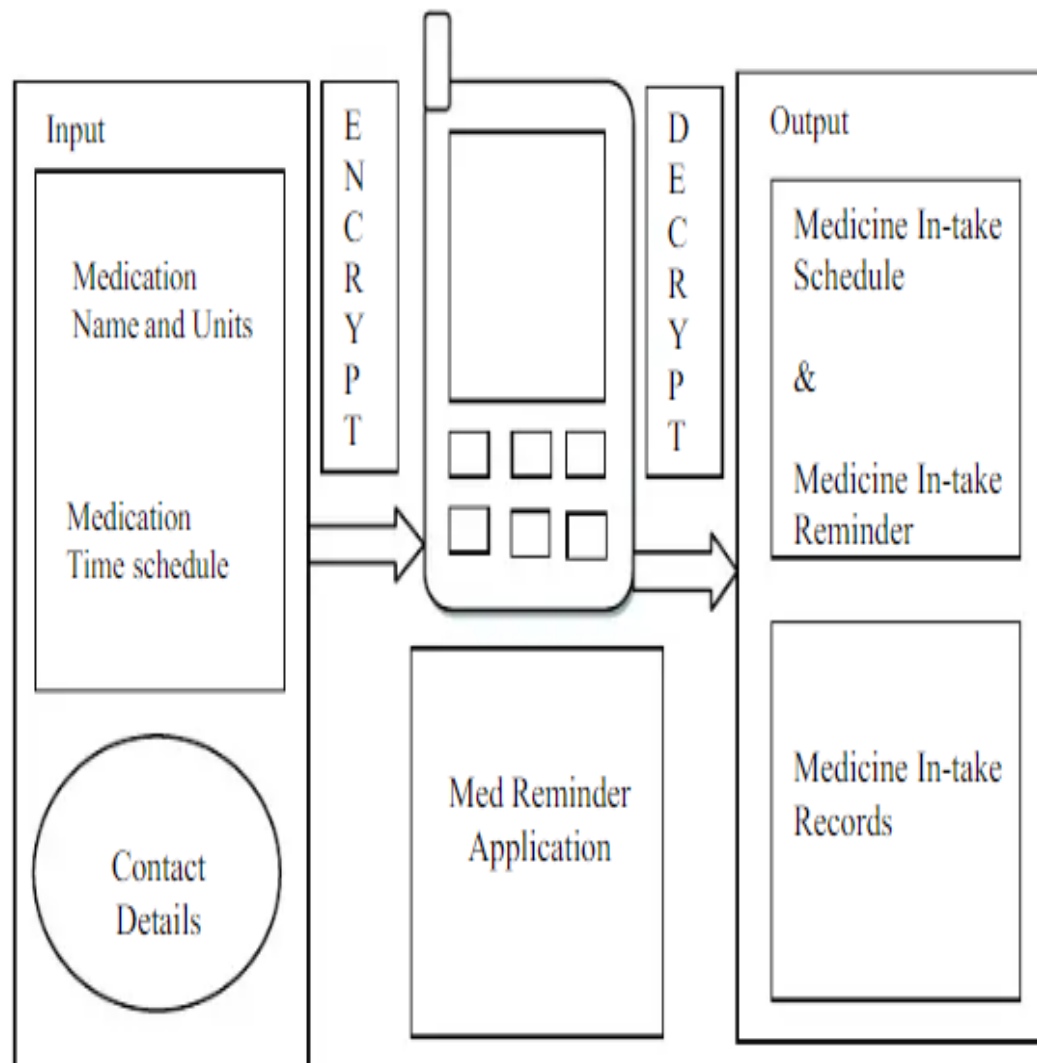


Figure 3.2: Input and output data flows of the system

# Chapter 4

## TESTING

Testing is an infinite process of comparing the invisible to the ambiguous in order to avoid the unthinkable happening to the anonymous. Testing provides developers a gain in confidence on the modules developed and ease in maintaining them. In this project, the testing is divided into three modules- Unit Testing, Integration Testing, and System Testing.

### 4.1 Unit testing

UNIT TESTING is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a WhiteBox testing technique that is usually performed by the developer. Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing. Unit Testing is important because software developers sometimes try saving time doing

minimal unit testing and this is myth because inappropriate unit testing leads to high cost Defect fixing during System Testing, Integration Testing and even Beta Testing after application is built. If proper unit testing is done in early development, then it saves time and money in the end.

Use Case: Splash Screen

Expected Outcome: The splash screen should open immediately after the application starts.

Actual Outcome: The splash screen opens immediately after the application starts.

Use Case: Hamburger Menu

Expected Outcome: The Hamburger menu should open immediately after sliding the screen or clicking on the hamburger icon and should not affect the current activity present on the screen.

Actual Outcome: The Hamburger menu opens immediately after sliding the screen or clicking on the hamburger icon and does not affect the current activity present on the screen.

Use Case: Add user activity

Expected Outcome: The details entered should be validated and no invalid entries should be accepted from the users.

Actual Outcome: The users receive and prompt error displaying the invalid detail entered by them and are asked to re-enter the value before saving the details.

Use Case: Edit User Activity

Expected Outcome: The activity should display the current details of the user and re-validate them once user requests to edit them.

Actual Outcome: The activity displays the current details of the user and re-validates them once user requests to edit them.



Use Case: Add Medicine Activity - Medicine Details

Expected Outcome: The users can enter the medicine details and dosage along with the image for the medicines.

Actual Outcome: The users can easily enter the medicine details and dosage along with the image for the medicines.

Use Case: Add Medicine Activity - Calendar Details

Expected Outcome: The users can enter the start and end date for the respective medicines and the end date should not be less than the start date.

Actual Outcome: The users can enter the start and end date for the respective medicines and will be prompted to re-enter the dates if the end date is less than the start date.

Use Case: Add Medicine Activity - Time Duration

Expected Outcome: The time duration should be of two types- daily and weekly. Weekly time duration should display the week details so that user can select a particular day and Daily should display the time which is to be reminded daily.

Actual Outcome: The time duration displays the two types- daily and weekly. “Weekly” time duration displays the week details so that user can select a particular day and “Daily” displays the time which is to be reminded daily.

Use Case: Calendar Activity

Expected Outcome: This activity should display all the details of the medicines categorized on the basis of day.

Actual Outcome: This activity displays all the details of the medicines categorized on the basis of day.

Use Case: Settings Activity- Enable/Disable Splash Screen

Expected Outcome: This activity should help the user to enable or disable the splash screen of the application.

Actual Outcome: This activity helps the user to enable or disable the splash screen of the application.

Use Case: Settings Activity - Turn On/Off Notifications

Expected Outcome: This activity should help the user to turn on/off the notifications.

Actual Outcome: This activity helps the user to turn on/off the notifications.

Use Case: Settings Activity - Delete User Profile

Expected Outcome: This activity should help the user to delete the user profile.

Actual Outcome: This activity helps the user to delete the user profile.

Use Case: Settings Activity - Change User Details

Expected Outcome: This activity should call the edit user details activity.

Actual Outcome: This activity calls the edit user details activity.

Use Case: Settings Activity - Reset Settings

Expected Outcome: The user should be prompted again before deleting all the data of the application and once user agrees, should delete all the data of the application.

Actual Outcome: The user is prompted again before deleting all the data of the application and once user agrees, deletes all the data of the application.

Use Case: Settings Activity - Delete Medication

Expected Outcome: The user can easily delete all the details of a specific medicine.

Actual Outcome: The user can easily delete all the details of a specific medicine and the changes will be updated in the database.

Use Case: Notification Use Case

Expected Outcome: The application notifications should contain sound, vibra-

tion, flashlight and screen flickering. It should work on all the android versions and should handle all the exceptions i.e. if user has a faulty hardware device like vibration, flashlight etc.

Actual Outcome: The application notifications contain sound, vibration, flashlight and screen flickering. It works on all the android versions(Android Oreo and above have a channel) and should handles all the exceptions i.e. if user has a faulty hardware device like vibration, flashlight etc.

Use Case: Database Creation - User

Expected Outcome: Complete Details of the user should be stored in the database. Also, the application should update and delete the stored details.

Actual Outcome: Complete Details of the user are saved in the database. Also, the application will update and delete the stored details.

Use Case: Database Creation - Medicines

Expected Outcome: The medicine table should store have all the details of the medicines along with the time and date. The users can easily modify and delete the details from the database table.

Actual Outcome: The medicine table stores all the details of the medicines along with the time and date. The users can easily modify and delete the details from the database table.

## 4.2 Integration testing

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units. Integration testing is one of the agile methodologies of software testing where

individual components or units of code are tested to validate interactions among different software system modules. In this process, these system components are either tested as a single group or organized iteratively.

Typically, system integration testing is taken up to validate the performance of the entire software system as a whole. The main purpose of this testing method is to expand the process and validate the integration of the modules with other groups. It is performed to verify if all the units operate in accordance with their specifications defined.

In Software testing, it is important that every system component gets integrated with the different application modules. Though each of the software modules is unit tested, there are critical chances for the modules to have defects, and this calls for the need for Integration testing to come into place. Significantly, system integration testing is taken up to effectively verify the various interactions between modules and also verifies the low-level and high-level software requirements as given in the software requirements specifications document.

Use Case: Add Medicine and Home Activity Integration

Expected Outcome: The add medicine activity details should be displayed on the home page.

Actual Outcome: The add medicine activity details are displayed on the home page.

Use Case: Calendar and Database Integration

Expected Outcome: The calendar should easily display all the details of the medicines stored in the database.

Actual Outcome: The calendar displays all the details stored in the database.

Use Case: Hamburger Integration

Expected Outcome: The Hamburger menu should help the user to call all the modules of the application and should contain all the exceptions.

**Actual Outcome:** The Hamburger menu will help the user to call all the modules of the application and contains all the exceptions.

**Use Case:** Add User and Edit User Integration

**Expected Outcome:** The details stores in the add user activity should be displayed in the edit user activity and the add user activity should be disabled after the user has been added.

**Actual Outcome:** The details stores in the add user activity are displayed in the edit user activity and the add user activity is disabled after the user has been added.

**Use Case:** Settings Activity Integration

**Expected Outcome:** All the functions in the settings activity should easily call the respective activities and functions.

**Actual Outcome:** All the functions in the settings activity easily call the respective activities and functions.

**Use Case:** Notification Integration

**Expected Outcome:** The notifications should be called from the background functions to remind medicines.

**Actual Outcome:** The notifications are called from the background functions to remind medicines.

## 4.3 System testing

System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of soft-

ware to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.

To check the end-to-end flow of an application or the software as a user is known as System testing. In this, we navigate (go through) all the necessary modules of an application and check if the end features or the end business works fine, and test the product as a whole system.

Use Case: Data Entered by Users

Expected Outcome: The data entered by users should easily stored the data should be used by the all the functions present is the application.

Actual Outcome: The data entered by users is easily stored the data will be used by the all the functions present is the application.

Use Case: Medicine Reminder

Expected Outcome: The application should provide the reminder notification for all the medicines at the specified time.

Actual Outcome: The application provides the reminder notification for all the medicines at the specified time.

Use Case: Missed Medication

Expected Outcome: The application should handle the missed medicine notifications.

Actual Outcome: The application stores the data for the missed medicines and the specified functions evaluate the results.

Use Case: Stability Check

Expected Outcome: The application should never crash and all the test cases should pass the results.

Actual Outcome: All the exceptions handled and all the bugs were fixed in the application.

# Chapter 5

## MODULES

There are six modules in this project. They are:

### **1) Profile creation**

We can add new user here with their name and contact details. Multiple users are allowed here. The details entered should be validated and no invalid entries should be accepted from the users. The users receive and prompt error displaying the invalid detail entered by them and are asked to re-enter the value before saving the details.

### **2) Medicine Activity**

The users can enter the medicine details and dosage along with the image for the medicines. The users can enter the start and end date for the respective medicines and will be prompted to re-enter the dates if the end date is less than the start date. The time duration displays the two types- daily and weekly. “Weekly” time duration displays the week details so that user can select a particular day and “Daily” displays the time which is to be reminded daily.



### **3) Notification Activity**

The application notifications contain sound, vibration, flashlight and screen flickering. It works on all the android versions(Android Oreo and above have a channel) and should handles all the exceptions i.e. if user has a faulty hardware device like vibration, flashlight etc.

### **4) Settings Activity**

This activity should help the user to enable or disable the splash screen of the application and helps the user to turn on/off the notifications. This activity helps the user to delete the user profile, Change User Details, Reset Settings and Delete medication.

### **5) Calender Activity**

This activity displays all the details of the medicines categorized on the basis of day and all the details stored in the database are displayed.

### **6) Database Creation**

Complete Details of the user are saved in the database. Also, the application will update and delete the stored details. The medicine table stores all the details of the medicines along with the time and date. The users can easily modify and delete the details from the database table.

## **Chapter 6**

### **RESULT ANALYSIS**

The application gives reliable reminders, good user interface, nice user experience and it supports many new features supporting medication adherence. The combination of all the functionalities provided is useful to the people of all ages. People of the greater ages are more likely to forget the medicine timings as well as remembering their appointments.

The users will get the schedule of medicine in-take time with medicine description, starting and ending date of medicine, notification, automatic alarm ringing system and the prescription is stored in the application. The automatic alarm ringing feature was proved beneficial to 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.

So this feature was found useful to the youths. Hence the overall system served well in our survey and it truly supports Medication Adherence.

## Chapter 7

### CONCLUSION

Many Medication Reminder Systems have been developed on different platforms. 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 adherence. Medication non-adherence reduces the effectiveness of a treatment and imposes a financial burden on health care systems.

The patients 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. The scheduled reminder will not suggest any kind of medicine which is not prescribed by the doctor that will assure the safety of the patient and also will avoid wrong dosages.

The medicine reminder will be very helpful to many patients. It helps to take proper medicine at right time. The cost of production is low as compared to other problem solutions. Health is the only thing that we should care the most. COVID-19 made all the people aware about the importance of health. In such a situation, forgetting about the medicine timings can severely affect the health of an

individual. Taking this into consideration, I came up with the idea of this project. The main feature of this project is that it is multi-user. so that all the family members especially the elder people who generally don't have smartphones can get benefited by saving their schedule in one of the smartphone in their family.

The application has been designed for users which are seated on chair or sofa, relaxing at their home, or on-the-go. As Long as the user has their medication on them, which they should, they will be able to use the application effectively. If the user's state of mind is busy, bored or angry, then such users will not enjoy the application. This application is expected to be used when the user receives a notification about their medication, when the user wants to enter medication information, or when they would like to see when they need to administer their next medication.

# Chapter 8

## APPENDIX

### 8.1 CODING

#### 8.1.1 AddNewMedBusinessLayer

```
package com.example.devan.remedaily.businesslayer;
import android.os.AsyncTask;
import android.support.annotation.NonNull;
import com.example.devan.remedaily.datalayer.AppDatabase;
import com.example.devan.remedaily.datalayer.Med;
import com.example.devan.remedaily.datalayer.User;
import java.util.ArrayList;
import java.util.List;
public class AddNewMedBusinessLayer {
    public static void AddMeds(@NonNull final AppDatabase db,
    int tagDaily, String medName,String dosage, String imagePath,
    String startDate, String endDate, ArrayList<ArrayList<String>> arrTimeItem) {
        AddMeds task = new
        AddMeds(db, tagDaily, medName, dosage, imagePath,
        startDate, endDate, arrTimeItem);
```

```
        task.execute();
    }
    private static boolean InsertMeds(@NonNull final AppDatabase
db, int tagDaily, String medName,
String dosage, String imagePath, String startDate,
String endDate, ArrayList<ArrayList<String>> arrTimeItem)
throws Exception {
        Med med = new Med();
        med.tagDaily = tagDaily;
        med.medName = medName;
        med.dosage = dosage;
        med.imagePath = imagePath;
        med.startDate = startDate;
        med.endDate = endDate;
        med.timeObject = arrTimeItem;
        db.medModel().insertMeds(med);
        return true;
    }
    public static List<Med> GetMedData(@NonNull final AppDatabase
db) { List<Med> medList = db.medModel().loadAllMeds();
        return medList;
    }
    public static String GetEmailID(@NonNull final AppDatabase
db){ List<User> userList = db.userModel().loadAllUsers();
        return userList.get(0).emailID;
    }
    private static class AddMeds extends AsyncTask<String,
String, String> {

        private final AppDatabase mDb;
```

```
private final int tagDaily;
private final String medName, dosage,
imagePath, startDate, endDate;
private final ArrayList<ArrayList<String>> arrTimeItem;
AddMeds(AppDatabase db, int tagDaily, String medName,
String dosage, String imagePath, String startDate,
String endDate, ArrayList<ArrayList<String>>
arrTimeItem) {
    mDb = db;
    this.tagDaily = tagDaily;
    this.medName = medName;
    this.dosage = dosage;
    this.imagePath = imagePath;
    this.startDate = startDate;
    this.endDate = endDate;
    this.arrTimeItem = arrTimeItem;
}

@Override
protected String doInBackground(final String... params) {
    try {
        InsertMeds(mDb, tagDaily, medName, dosage,
        imagePath, startDate, endDate, arrTimeItem);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
}
```

### 8.1.2 HamburgerBusinessLayer

```
package com.example.devan.remedaily.businesslayer;
import android.support.annotation.NonNull;
import com.example.devan.remedaily.datalayer.AppDatabase;
import com.example.devan.remedaily.datalayer.User;
import java.util.List;
public class HamburgerBusinessLayer {
    public static StringBuilder ShowUserInfo(AppDatabase appData){
        StringBuilder sb = new StringBuilder();
        List<User> Users = ShowUsers(appData);
        for (User User : Users) {
            sb.append(String.format(
                "%s %s\n", User.firstName , User.lastName));
        }
        if (sb.length()==0){
            sb.append("User");
        }
        return sb;
    }
    private static List<User> ShowActiveUser(AppDatabase db){
        List<User> med1 = db.userModel().loadAllUsers();
        return med1;
    }
    private static List<User> ShowUsers(@NonNull final
AppDatabase db) {
        return ShowActiveUser(db);
    }
}
```



### **8.1.3 UserDetailsBusinessLayer**

```
package com.example.devan.remedaily.businesslayer;
import android.os.AsyncTask;
import android.support.annotation.NonNull;
import com.example.devan.remedaily.datalayer.AppDatabase;
import com.example.devan.remedaily.datalayer.User;
public class UserDetailsBusinessLayer {
    public static boolean IsUserPresent(AppDatabase db){
        boolean flag =db.userModel().userPresent();
        return flag;
    }
    public static void InsertRecordsAsync(@NonNull final
AppDatabase db, String FirstName, String LastName,
String Age,String EmailID) throws Exception {
        InsertRecords task = new InsertRecords(db,
        FirstName, LastName, Age,EmailID);
        task.execute();
    }
    public static boolean InsertSyncUser(@NonNull final
AppDatabase db, final String FirstName,
final String LastName, final String Age,final String
EmailID) throws Exception {
        if (!FirstName.isEmpty()&&!LastName.isEmpty()
&&!Age.isEmpty()&&!EmailID.isEmpty()){
            return InsertWithTestDataUser(db, FirstName,
            LastName, Age,EmailID);
        }
        else
            return false;
    }
}
```

```
private static boolean InsertWithTestDataUser(AppDatabase
db, final String FirstName,
final String LastName, final String Age,final String
EmailID) throws Exception {
    db.userModel().deleteUser();
    return AddUser(db, FirstName, LastName, Age,EmailID);
}
private static boolean AddUser(final AppDatabase db,
final String FirstName, final String LastName,
final String Age,final String EmailID) throws Exception{
    User user = new User();
    user.firstName = FirstName;
    user.lastName = LastName;
    user.age = Age;
    user.emailID= EmailID;
    user.userPresent = true;
    if (1 == db.userModel().insertUser(user)) {
        return true;
    } else {
        return false;
    }
}
private static class InsertRecords extends AsyncTask
<String, String, String> {
    private final AppDatabase mDb;
    private final String FirstName, LastName, Age,EmailID;
    InsertRecords(AppDatabase db, String FirstName,
String LastName, String Age,String EmailID) {
        mDb = db;
        this.FirstName = FirstName;
```

```
        this.LastName = LastName;
        this.Age = Age;
        this.EmailID= EmailID;
    }
    @Override
    protected String doInBackground(final String ...
params) {
        try {
            InsertSyncUser (mDb, FirstName ,LastName ,
            Age,EmailID );
        } catch (Exception e) {
            e.printStackTrace ();
        }
        return null;
    }
}
```

#### 8.1.4 ResetApplicationLayer

```
package com.example.devan.remedaily.businesslayer;
import com.example.devan.remedaily.datalayer.AppDatabase;
public class ResetApplicationLayer {
    public static void resetUserData(final AppDatabase db)
    {
        db.userModel().deleteUser();
    }
    public static void resetMedData(final AppDatabase db)
    {
        db.medModel().deleteAll();
    }
}
```

```
    }  
}
```

### 8.1.5 DisplayNotification

```
package com.example.devan.remedaily.View;  
import android.annotation.TargetApi;  
import android.app.Notification;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.ContextWrapper;  
import android.content.Intent;  
import android.graphics.Color;  
import android.os.Build;  
import android.support.annotation.RequiresApi;  
import android.support.v4.app.NotificationCompat;  
import com.example.devan.remedaily.R;  
import static android.provider.Settings.System.  
DEFAULT_NOTIFICATION_URI;  
class DisplayNotification extends ContextWrapper {  
    public DisplayNotification(Context base) {  
        super(base);  
    }  
    @TargetApi(Build.VERSION_CODES.M)  
    @RequiresApi(api = Build.VERSION_CODES.M)  
    public void createNotification(String messageDetails ,  
        String descriptionDetails) {  
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
```

```
String id = "my_channel_01";
CharSequence name = "Remedaily";
int importance = NotificationManager.IMPORTANCE_HIGH;
NotificationChannel mChannel = null;
mChannel = new NotificationChannel(id, name,
importance);
mChannel.setDescription("uses notifications
access for ring and vibrate.");
mChannel.enableLights(true);
mChannel.setLightColor(Color.RED);
mChannel.setSound(mChannel.getSound(), mChannel.
getAudioAttributes());
mChannel.enableVibration(true);
mChannel.setVibrationPattern(new long[]{100, 200,
300,400, 500, 400, 300, 200, 400});
NotificationManager mNotificationManager
= (NotificationManager) getSystemService
(Context.NOTIFICATION_SERVICE);
mNotificationManager.createNotificationChannel
(mChannel);
String channelId = "my_channel_01";
Notification notification = new
Notification.Builder(getApplicationContext())
.setContentTitle("Medicine: "+messageDetails)
.setContentText(descriptionDetails+"
Click to view details")
.setSmallIcon(R.drawable.hands)
.setSound(DEFAULT_NOTIFICATION_URI)
.setChannelId(channelId).setAutoCancel(true)
.setSound(mChannel.getSound(), mChannel
```

```
.getAudioAttributes()).build();
notification.contentIntent= PendingIntent.
getActivity(this, 0,new.Intent(this,Notification.
class).putExtra("name",messageDetails).putExtra
("description",descriptionDetails),PendingIntent
.FLAG_CANCEL_CURRENT);
mNotificationManager.notify(001, notification);

}
else {
    Notification notification =
    new NotificationCompat.Builder(this.
    getApplicationContext()).setSmallIcon
    (R.drawable.hands).setContentTitle("Medicine:" +
    messageDetails).setContentText
    (descriptionDetails+" Click to view details")
    .setAutoCancel(true).setVibrate(new long[]
    {100, 200, 300, 400, 500, 400, 300, 200, 400})
    .setSound(DEFAULT_NOTIFICATION_URI).build();
    notification.contentIntent= PendingIntent.
    getActivity(this, 0,newIntent(this,Notification
    .class).putExtra("name",messageDetails).putExtra
    ("description",descriptionDetails),PendingIntent
    .FLAG_CANCEL_CURRENT);
    NotificationManager mNotificationManager=Notification
    SystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(001, notification);
}
EnableFlashLight enableFlashLight = new
EnableFlashLight(this);
```

```
        enableFlashLight.enableFlash();
    }
}
```

### **8.1.6 SplashScreen**

```
package com.example.devan.remedaily.View;
import android.content.Intent;
import android.os.Build;
import android.support.annotation.RequiresApi;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import com.example.devan.remedaily.R;
public class SplashScreen extends AppCompatActivity {
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen);
        Thread thread = new Thread(){
            @Override
            public void run() {
                try {
                    if( SettingsDetailsFetch.splashScreenOnOff
                        (getApplicationContext()))
                    {
                        sleep(1500);
                    }
                    Intent intent = new Intent(getApplicationContext());
                    startActivity(intent);
                }
            }
        };
        thread.start();
    }
}
```

```
        SplashScreen.this.finish();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
};
thread.start();
}
}
```

### 8.1.7 NotificationDisplayMedicine

```
package com.example.devan.remedaily.View;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.MenuItem;
import android.widget.TextView;
import com.example.devan.remedaily.R;
public class NotificationDisplayMedicine extends
AppCompatActivity {
    TextView title;
    TextView medName;
    TextView medDosage;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(R.layout.display_current_medicine);
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        Bundle extras = getIntent().getExtras();
```



```
        title=findViewById(R.id.dayTv);
        medName=findViewById(R.id.medicineNameTv);
        medDosage=findViewById(R.id.medicineDosageTv);
        title.setText("Medicine Details");
        medName.setText(extras.getString("name"));
        medDosage.setText(extras.getString("description"));
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                onBackPressed();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

### **8.1.8 MainActivity**

```
package com.example.devan.remedaily.View;
import android.app.LauncherActivity;
import android.content.Intent;
import android.os.Bundle;
import android.provider.Settings;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import com.example.devan.remedaily.R;

public class MainActivity extends AppCompatActivity {
    private TextView txtView;
    public Button userDetailsBtn , editUserDetailsBtn , calenderBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtView= findViewById(R.id.txtView);
        userDetailsBtn=findViewById(R.id.userDetailsBtn);
        editUserDetailsBtn=findViewById(R.id.editUserDetailsBtn);
        userDetailsBtn.setOnClickListener(new View.
            OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent=
                        new Intent(MainActivity.this , UserDetails.class);
                    startActivity(intent);
                }
            }
        );
    }
}
```

```
});
editUserDetailsBtn.setOnClickListener(new View.
OnClickListener() {

    @Override
    public void onClick(View v) {
        Intent intent=
        new Intent(MainActivity.this ,EditUserDetails
        .class);
        startActivity(intent);
    }
});
}

}
```


## 8.2 SCREENSHOTS



Figure 8: Home Screen


16:41 4G 0.00 KB/S LTE 40


← Add Medicine

 medicine name

medicine dosage

Daily ☐

Start Date  mm/dd/yyyy

End Date  mm/dd/yyyy

MO TU WE TH FR SA SU

SAVE ADD TIME

Figure 8.1: Add Medicine

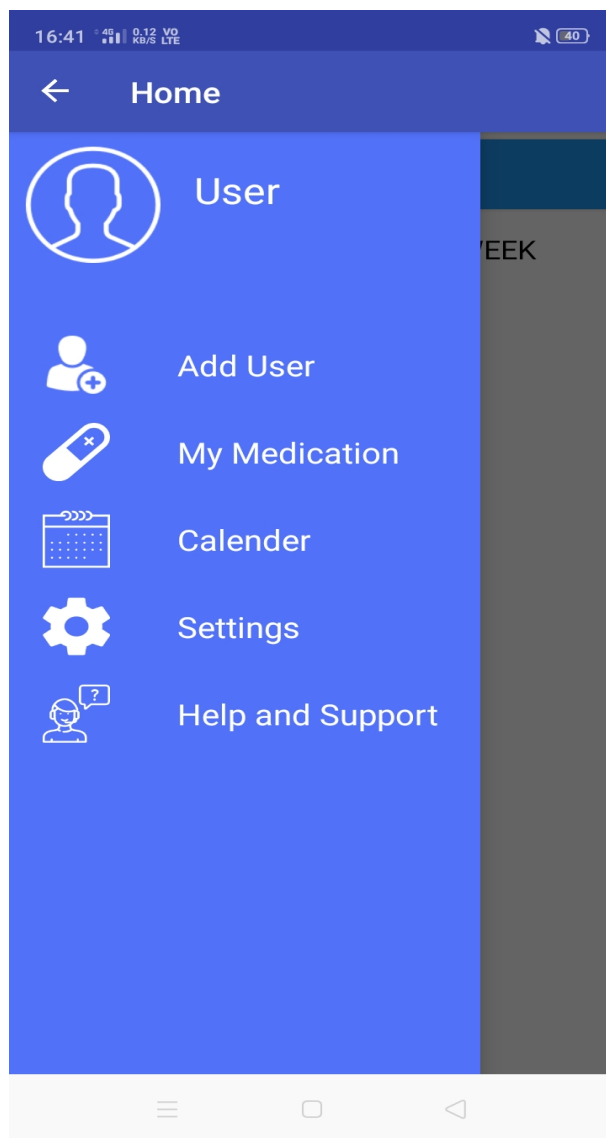
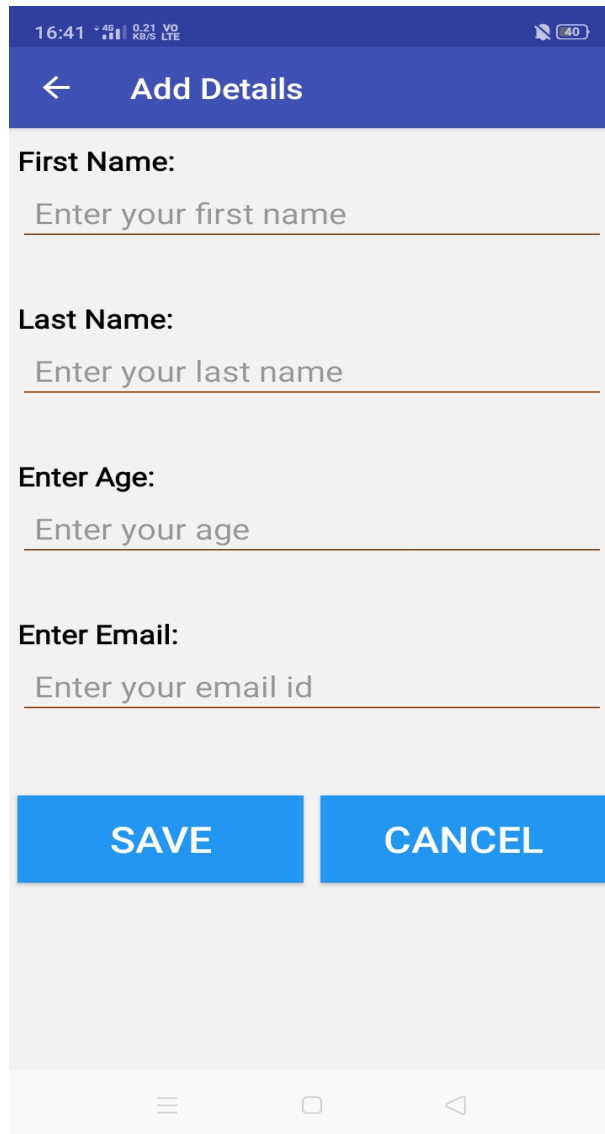


Figure 8.2: Home Screen Options



The image shows a mobile application interface for adding user details. At the top, a blue header bar contains a back arrow and the text "Add Details". Below this, the form is divided into four sections, each with a label and a text input field: "First Name:" with the placeholder "Enter your first name", "Last Name:" with "Enter your last name", "Enter Age:" with "Enter your age", and "Enter Email:" with "Enter your email id". At the bottom of the form are two blue buttons labeled "SAVE" and "CANCEL". The entire form is set against a light gray background. The top of the screen shows a status bar with the time "16:41", signal strength, data usage "0.21 KB/s", and battery level "40". The bottom of the screen features a white navigation bar with three icons: a hamburger menu, a square, and a back arrow.

Figure 8.3: Add User

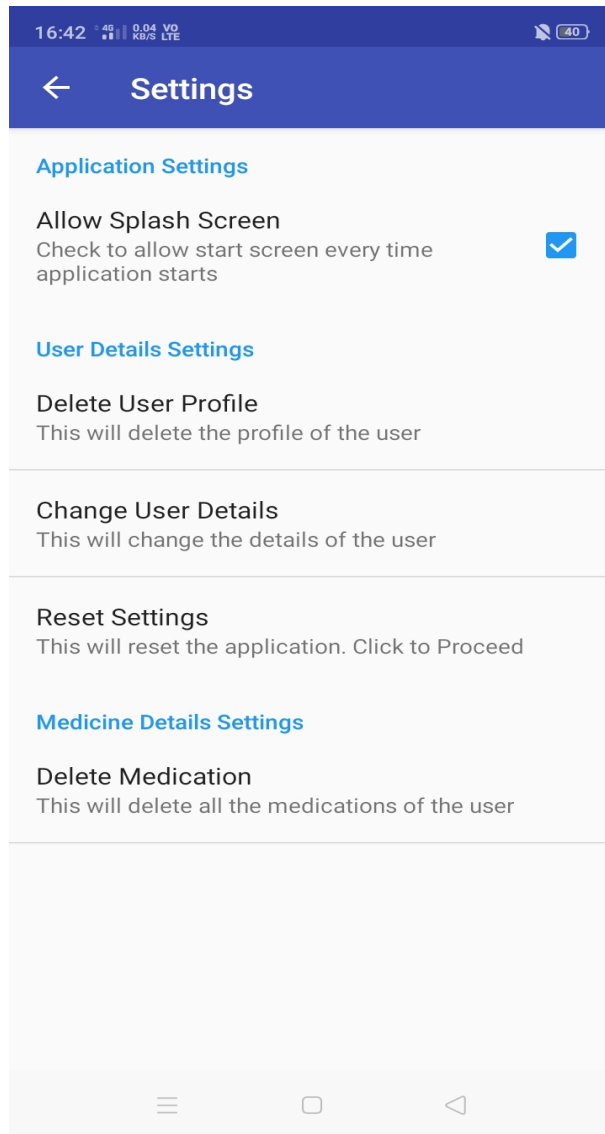


Figure 8.4: Settings



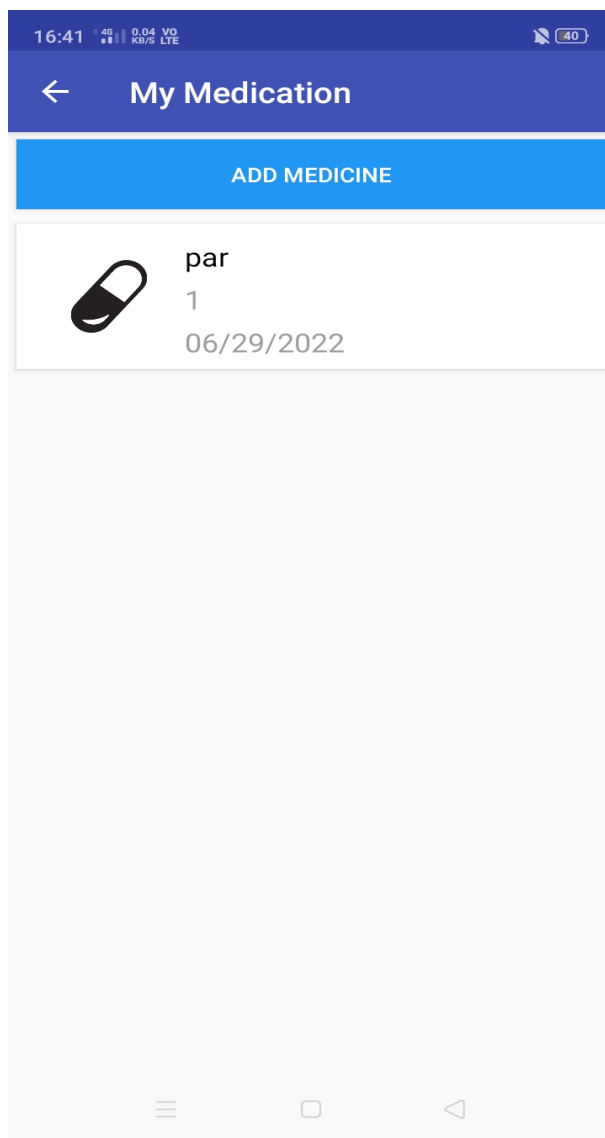


Figure 8.5: My Medication

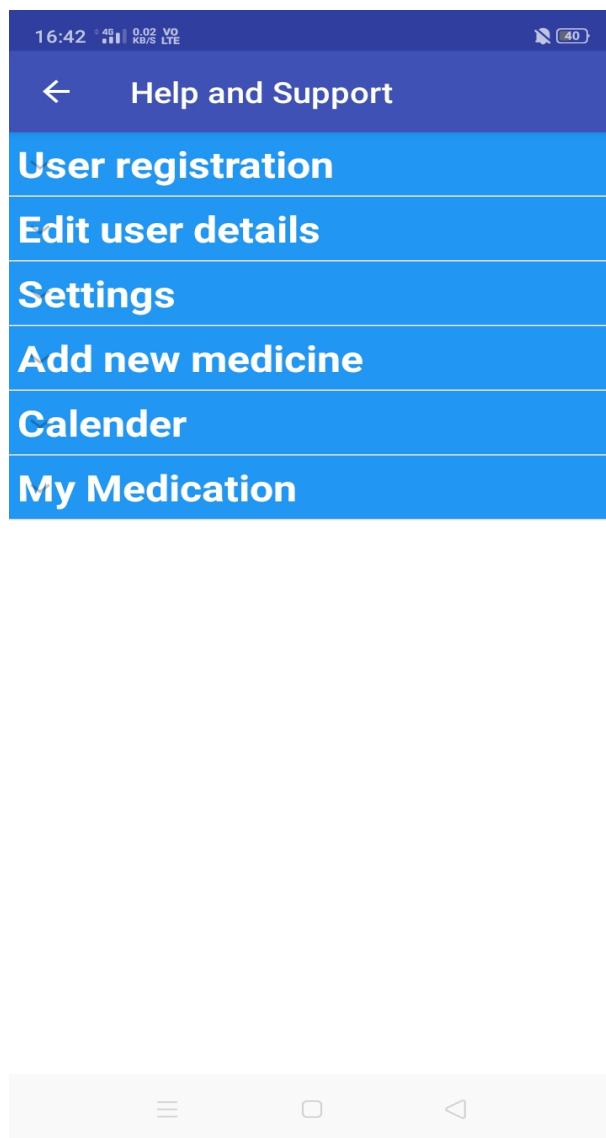


Figure 8.6: Help and Support

## Chapter 9

## REFERENCES

- <https://www.testingexcellence.com/best-software-testing-quotes/>
- <https://developer.android.com/topic/libraries/architecture/room>
- <http://developer.android.com/guide/topics/data/data-storage.html>
- <http://www.sqlite.org>.
- "Android security overview," <http://source.android.com/tech/security/index.html>.