



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF MEDIA AND EDUCATIONAL INFORMATICS

Android application for elderly people

Supervisor:

BAKONYI VIKTÓRIA

MSc mathematics-physics-informatics teacher

Author:

Diar Bojaxhi

Computer Science BSc

Budapest 2021

EÖTVÖS LORÁND UNIVERSITY
FACULTY OF INFORMATICS

Thesis Registration Form

Student's Data:

Student's Name: Bojaxhi Diar

Student's Neptun code: GAUTL5

Course Data:

Student's Major: Computer Science BSc

I have an internal supervisor

Internal Supervisor's Name: BAKONYI VIKTÓRIA

Supervisor's Home Institution:

ELTE IK Média és Oktatásinformatika Tanszék

Address of Supervisor's Home Institution:

Budapest, Pázmány Péter stny. 1/C., 1117

Supervisor's Position and Degree:

Master teacher, Msc mathematics-physics-informatics teacher

Thesis Title: Android application for elderly people

Topic of the Thesis:

(Upon consulting with your supervisor, give a 150-300-word-long synopsis of your planned thesis.)

For my thesis project, I will make an android application for the elderly people. Everyone needs additional care and attention as they age and elderly people are gradually becoming an increasingly large category of application users. Imagine an idea for an application that will make their lives easier. Many of them are required to take a number of medications throughout the day at specific times. And with this application that will be possible to maintain, making it easy for them to remember, and keep track.

I will add the following functionalities to the software:

- Users will be able to register their medication
- They will be able to track the usage of their medications
- There will be reminders when to take the medications/Push notifications
- Notes where they can write about their day
- To do's to keep their goals on track
- Tips on how to keep a more healthy life

Budapest, 2020.06.06.

Contents

1 Introduction.....	5
1.1 Solution to the problem	5
2 User Documentation.....	6
2.1 System requirements	6
2.2 Installing the application.....	6
2.3 Login Activity	7
2.3.1 Register email.....	8
2.3.2 Reset password	9
2.3.3 Main page.....	10
2.3.4 Account settings	11
2.3.5 Application Information	12
2.3.6 Register medications.....	12
2.3.7 Schedule medication.....	14
2.3.8 Notes	16
2.3.9 To do's	17
2.3.10 Healthy tips	18
2.3.11 Multi-language support.....	20
2.3.12 Dark mode.....	21
3 Developer Documentation.....	22
3.1 Creating the program	22
3.2 Program architecture	22
3.3 Use-Case Diagram.....	23
3.4 Program structure	24
3.5 Firebase.....	25
3.5.1 Firebase User Authentication	25
3.5.2 Firebase Realtime Database	27
3.5.3 Firebase Cloud Messaging	28
3.5.4 Firebase Crashlytics.....	28

3.6 Activities	29
3.7 MVVM Architecture	31
3.8 Room Database	32
3.9 CI - Jenkins	33
3.9.1 Connection Jenkins and GitHub	34
3.9.2 Build Job	34
3.10 Testing	36
3.10.1 UI and Unit Tests.....	37
4 Conclusion	39
4.1 Future work.....	39
Bibliography	40

Chapter 1

Introduction

Nowadays, technology has become an unavoidable need and necessity for all ages. Considering the fact that starting from mid-age almost everyone starts using medication and through years, the therapy is extended to the consumption of more medication.

It is common that people of 60+ years old have some medical therapies to undertake and life dynamics sometimes makes it difficult to track the proper consumption of the medication. Nowadays people deal with different situations and difficulties and therefore sometimes forget to take the medication or do necessary tests on time as prescribed by the doctor.

1.1 Solution to the problem

Smartphones are in the hands of all people even those of elderly ages and smartphones very often make things easier for all. We all have someone at home, like a grandfather or an aunt who are elderly and have different therapies to follow as prescribed by the doctor, and in many cases, they forget to take medication on time and therefore their health can deteriorate. In order to avoid such issues, I was motivated by the fact that careless habits can sometimes be fatal for the elderly and thought about the development of an application, which would be the personal assistant to elderly people to keep them alert about their health and live a quiet life without worrying for details.

Elderly people need an application, which is easily understandable and can be easily used by them. This application would be an assistant to manage their health by adding their medication to the application. It is normal for everyone to forget taking the medication or do what it takes to take care of personal health. This application would be a personal assistant which will organize daily activity and keep you organized for daily life. I strongly believe that this application would help every user stay healthy and organize better.

Chapter 2

User Documentation

To be able to use the application, users will first need to create an account.

After successfully creating the account there will be user authentication which restricts the application to only those users who are authenticated, once verification is done the user can go back to the application and log in with the registered credentials. User can reset their password in case they forget it or even change it to a different password if they wish so.

Using this application, they will be able to store their everyday medication intakes, track how much they have taken so far, schedule when they need to take a certain medication, write notes about their day, save to do's which they can connect with a date to it, meaning that will be the date which that activity will need to be accomplished and healthy tips to remind them what is healthy for their lifestyle, and what is not.

2.1 System requirements

In order to run the application, we need an Android Emulator [2], and for that, we need the following system requirements: Windows 7 or later (64-bit), macOS 10.10 (Yosemite) or higher (64-bit), Linux GNOME or KDE Desktop (64-bit) [2].

Besides, the computer needs to be more powerful with 4GB RAM minimum, 8GB RAM recommended, and CPU Intel i3, i5, i7, or AMD Multi-core processors. Another solution is you can run the application on a physical device without worrying about the CPU.

2.2 Installing the application

The first thing we need to do is going to the GitHub repository which I will attach the URL at the end of the paper [1]. After, we need to clone the repository to the computer. After cloning, considering we already have Android Studio installed, we open Android Studio and open the

application files which we cloned from GitHub. After having the application set up in Android Studio, we have to create a virtual device and we can do that from Tools -> AVD Manager -> Create Virtual Device, and then we need to pick which device we want. And now that we have Android Emulator, we can go ahead and run the application from Android Studio, Android Emulator will pop up eventually.

2.3 Login Activity

The first-time user opens the already installed application, the login page will appear.

There are two fields required to be filled in order to proceed inside the application, email address and a password, user can set visibility on or off, according to their preference ([Figure 1.1](#)). If the user is already registered the user can log in and it will redirect to the Signed In page. But in the case where the user is not registered, then registration is required beforehand.

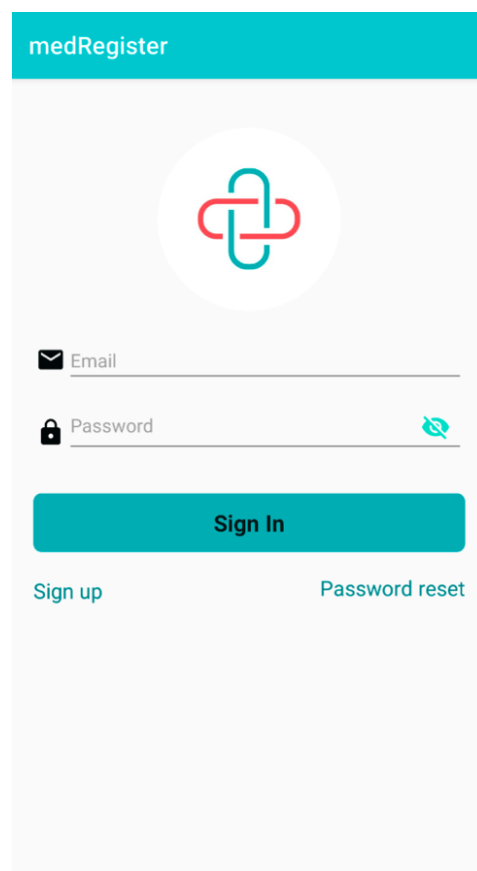
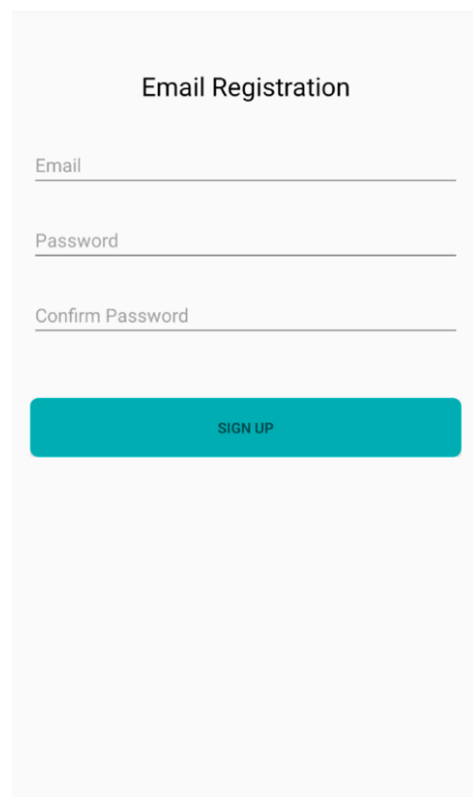


Figure 1.1 Login Activity

2.3.1 Register email

To be able to create a new account, all the required fields should be filled, email address with any email domain, a password which should have complexity, contain at least one digit, at least one lowercase letter, at least one uppercase letter, at least one special character and length of at least 8 characters and a maximum of 20, and for the last required field user needs to confirm the password ([Figure 1.2](#)). In the case where all the information is in the correct parameters, the user will be registered. After a verification link will be sent to the given email, and in order to finish the registration, the user needs to click on the link ([Figure 1.2.1](#)), which will then do the verification. After we are done with the verification, we may go back to the Login page and use our credentials we registered our account with. If everything is correct, the user will be redirected to the main page.

The image shows a web form titled "Email Registration". It contains three input fields: "Email", "Password", and "Confirm Password". Below these fields is a teal-colored button with the text "SIGN UP" in white capital letters.

Hello,

Follow this link to verify your email address.

https://medregister-a7344.firebaseio.com/_/auth/action?mode=verifyEmail&email=AlzaSyDBgEUBUHnxYK1czKiUylu30LVawsspSQU&lang=en

If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your project-799997579506 team

Figure 1.1.1 Verification Link Email

Figure 1.2 Register Activity

2.3.2 Reset password

User has the option to reset the password in case they forgot it, and they are not able to sign in into the account. After clicking on the password reset button, a dialog will appear ([Figure 1.3](#)) and the user needs to input their email address which was used for registration, if the email is correct and it exists, the user will get a reset password link to their email ([Figure 1.3.1](#)).

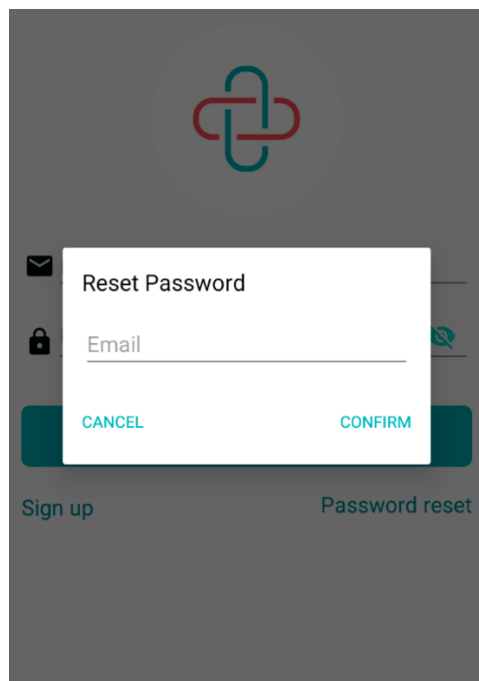


Figure 1.1 Reset password dialog

Hello,

Follow this link to reset your project-799997579506 password for your diar.bojaxhi@gmail.com account.

https://medregister-a7344.firebaseio.com/_/auth/action?mode=resetPassword&oobCode=CwcCvESbqAlzaSyDBgEUBUHnxYK1czKiUylu30LVawsspSQU&lang=en

If you didn't ask to reset your password, you can ignore this email.

Thanks,

Figure 1.3.1 Reset password URL

2.3.3 Main page

After the user is finally logged in to his account, the main page containing the activities will come up next (Figure 1.4). There are five different activities as listed, register medication activity, schedule medication activity, notes activity, to-do activity and healthy tips activity. If the user clicks in one of those it will send them to the corresponding activity. On the main page, we also have a drop-down menu on the right-hand side, with three options (Figure 1.4.1). The first option is Account Settings, the second option is Information about the application and the last option is Sign Out of the application. If the user decides to sign out, a dialog will appear asking for confirmation if the user wants to sign out or stay logged in.

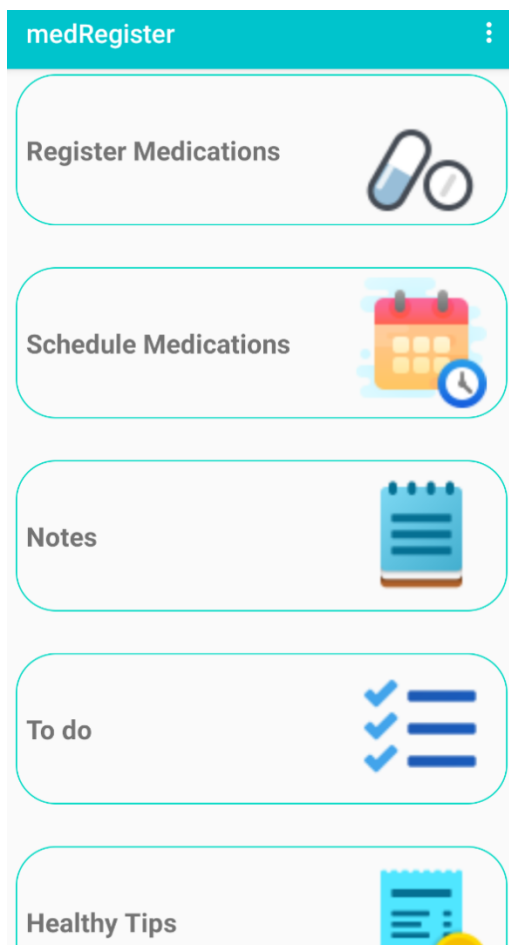


Figure 1.4 Home page

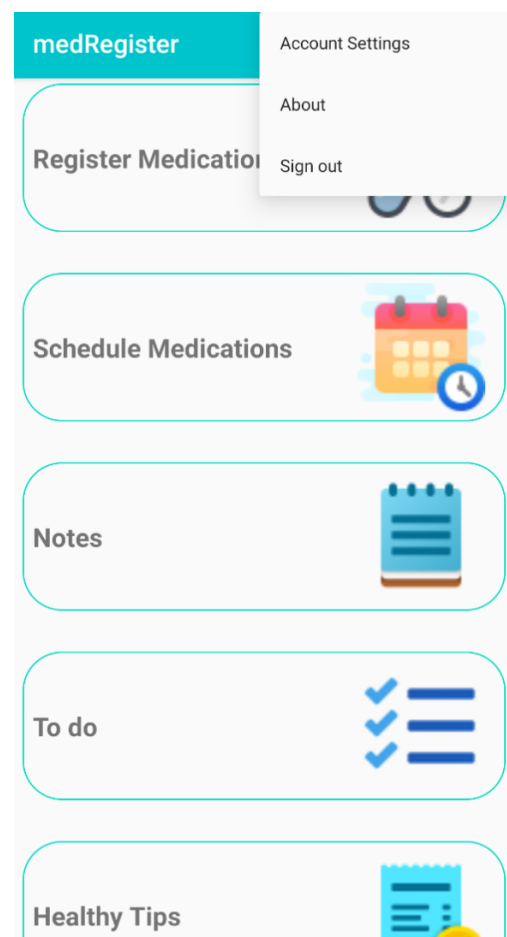
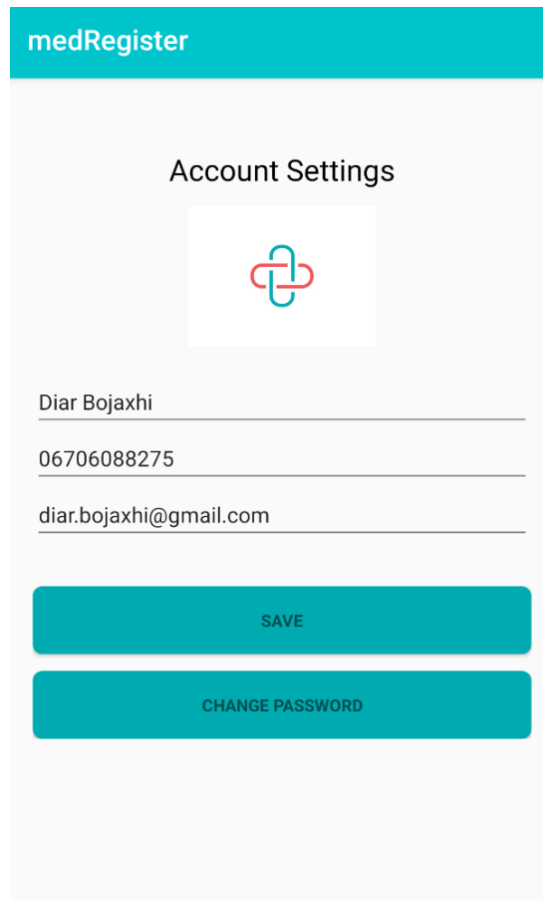


Figure 1.1.1 Menu


2.3.4 Account settings

Account settings are located under the drop-down menu on the right-hand side of the main page. After initiating we get the following layout, where user can input their name, and phone number, which is editable, after typing the information on the fields, in order to save it, the button save needs to be clicked. Email address is not editable, as the field shows the current signed in email address. There is a possibility of changing the password from inside the application, all we need to do is click on the Change password button, and a reset password link will be sent to the currently logged-in user email ([Figure 1.5](#)).



medRegister

Account Settings



Diar Bojaxhi

06706088275

diar.bojaxhi@gmail.com

SAVE

CHANGE PASSWORD

Figure 1.1 Account Settings

2.3.5 Application Information

Application information is located in the same menu as account settings; there we can find the information of the application, such as version code and version name of the currently installed application ([Figure 1.6](#)).

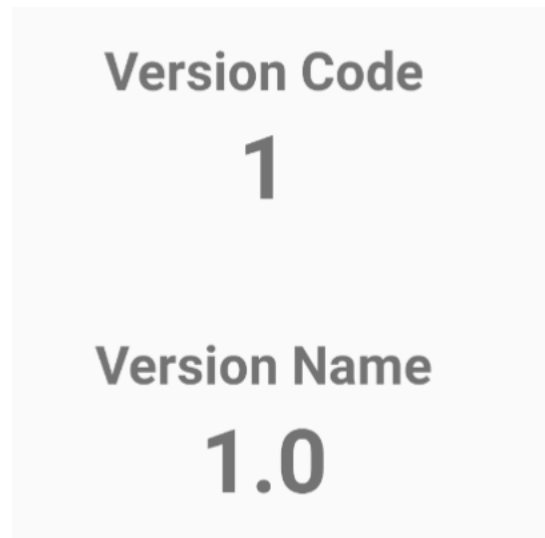
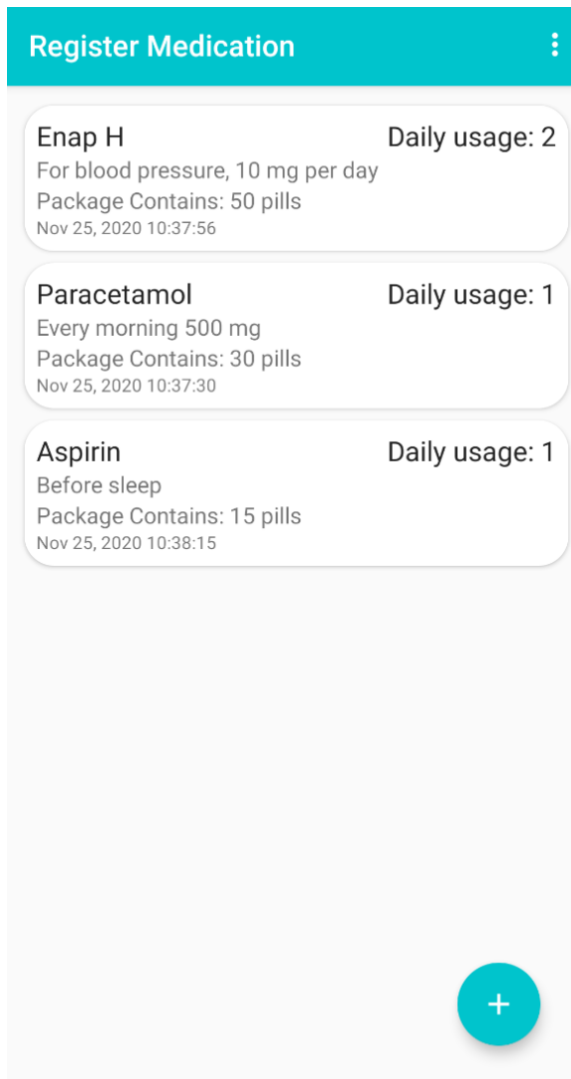


Figure 1.1 Application Information

2.3.6 Register medications

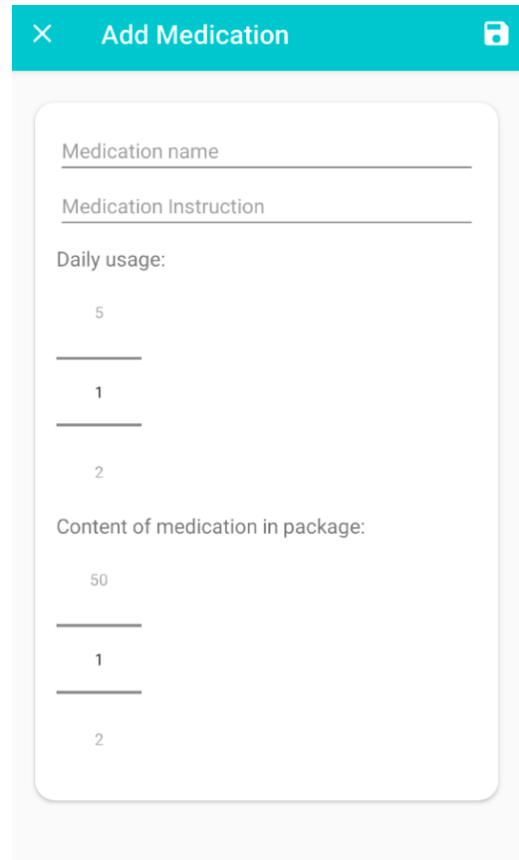
This activity will be useful for the user in order to register their everyday medications that they take and keep track of them as they consume them. So, first thing the user needs to do is, click on the add button ([Figure 1.7](#)), which will redirect the user to the next page, which will ask to fill the required fields, such as medication name, medication instruction, daily usage of how many times a day it is required to be taken and how much medication does the package contain ([Figure 1.7.1](#)). After filling the required fields, the user can save the medication by clicking on the save button on the top right-hand side. After saving the medication, the user is redirected back to the registers page, along with the other registered medications. In the case when the user is inside while adding the medication and does not wish to save the medication and wants to go back, on the top left-hand

side, there is a close button, which will bring the user back without saving the typed information. The user can keep track of how much medications they used, based on the creation time of the medication, this way they can go back and update the information, by clicking on the listed medication, which will redirect the user to the saved information, and if they change one of the fields, the time of creation will change to the updated time.

The 'Register Medication' screen features a teal header with the title and a close icon. It displays three medication entries, each in a white card with rounded corners. Each card contains the medication name, its purpose, dosage, package size, and creation timestamp, along with a 'Daily usage' count. A teal circular button with a white plus sign is located at the bottom right.

Medication Name	Daily usage	Details
Enap H	2	For blood pressure, 10 mg per day Package Contains: 50 pills Nov 25, 2020 10:37:56
Paracetamol	1	Every morning 500 mg Package Contains: 30 pills Nov 25, 2020 10:37:30
Aspirin	1	Before sleep Package Contains: 15 pills Nov 25, 2020 10:38:15

Figure 1.7 Register medication activity

The 'Add Medication' screen has a teal header with a close icon and a save icon. It contains a form with three sections: 'Medication name' and 'Medication Instruction' (both with text input fields), and 'Daily usage:' (with a numeric input field showing '5'). Below these is the 'Content of medication in package:' section, which includes a numeric input field showing '50' and two radio button options labeled '1' and '2'.

Medication name

Medication Instruction

Daily usage:

5

Content of medication in package:

50

1

2

Figure 1.1.1 Add medication activity

The user can delete the medications as well, by swiping right or left on the listed medication ([Figure 1.8](#)). Alternatively, in the case when they want to delete all medications at once ([Figure 1.8.1](#)), they can do so by opening the menu on the top right-hand side, which has the option to delete all medications.

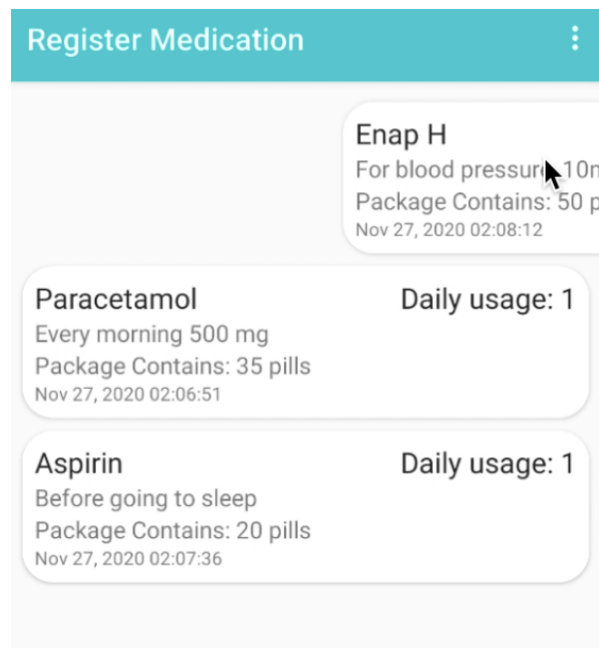


Figure 1.8 Swipe right or left to delete

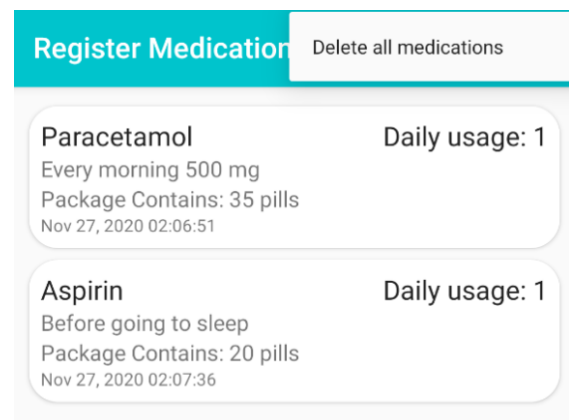


Figure 1.8.1 Menu option to delete all

2.3.7 Schedule medication

User can schedule medication intake at any specific time of the day they would like to be reminded. All user has to do, is click on the add schedule button ([Figure 1.9](#)), which will immediately redirect the user to the second page where they need to fill the required fields, such as medication name, pick a time from the time picker when clicking on the field, dose which would supposedly mean in mg unit. After the user has chosen the time and filled all the fields, they can save the scheduled medication by clicking on the save button on the top right-hand side ([Figure 1.9.1](#)). After saving the scheduled activity, the user will get redirected back to the scheduled medications listing, along

with the other scheduled activities. In the case when the user is inside while scheduling the activity and does not wish to go any further, on the top left-hand side, there is a close button, which will bring the user back without saving the rest of the information. The user can update the information as well, by clicking on the listed scheduled activity, which will redirect to the saved information, and they can change the medication name or the dose field, and again in order to save the updated information user needs to click on the save button on the top right-hand side.

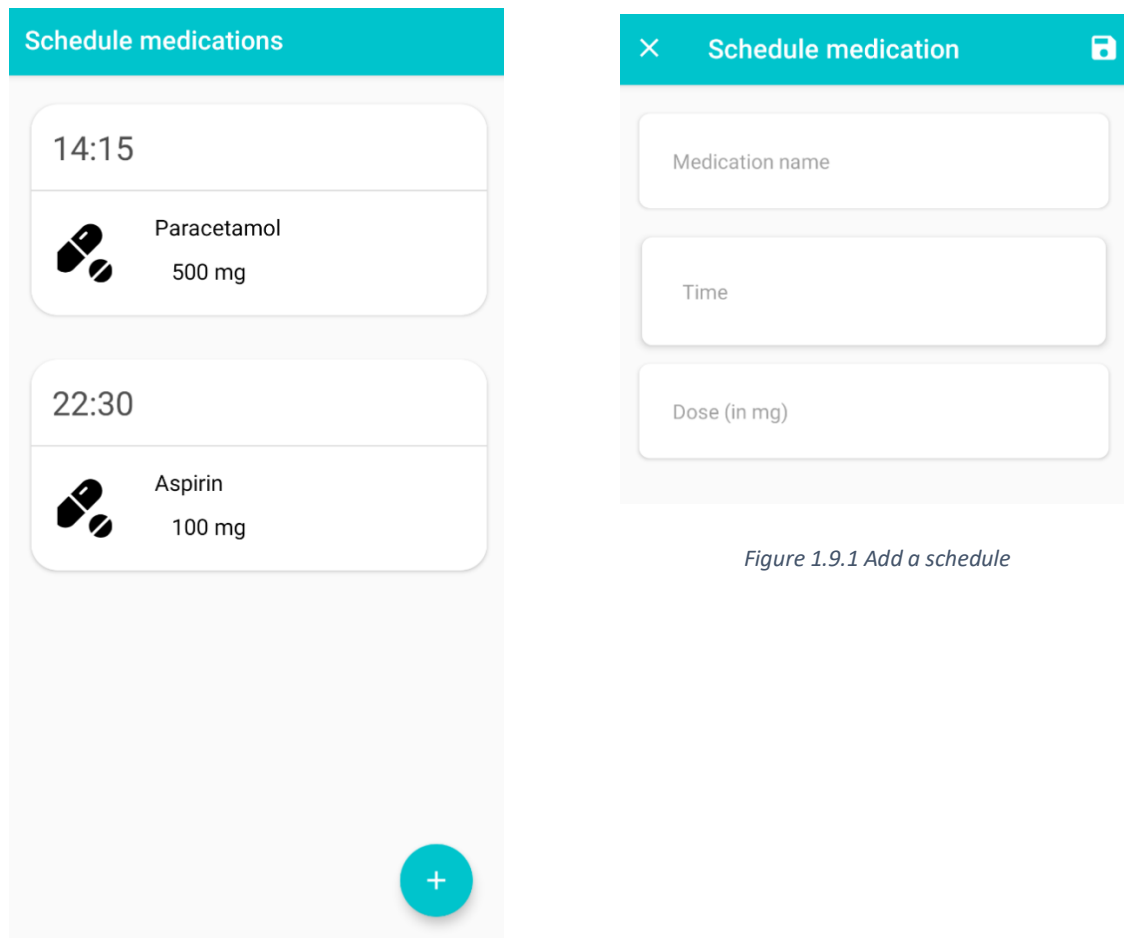


Figure 1.9.1 Add a schedule

Figure 1.9 Schedule medication activity

When the time of the scheduled activity comes, the user will receive a notification ([Figure 1.10](#)), saying the time for the medication intake has come. And when the user clicks on the notification, they will be redirected to the scheduled activities page, and notification will disappear from the notifications bar.

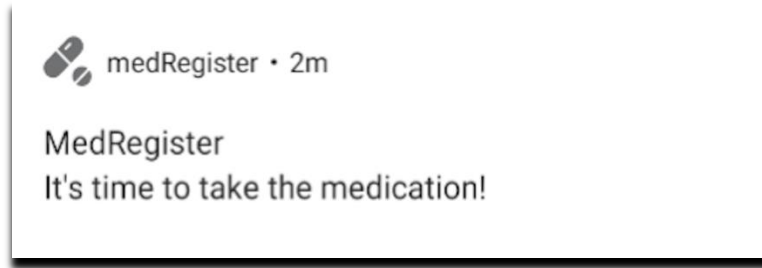


Figure 1.10 Notification from scheduled activity

2.3.8 Notes

Notes activity will be useful to the user in order for them to write down something that they want to do in the future, or something that they would like to have it written down, such as motivational sentences, or name of a movie which they would like to remember. Writing things down is a stress reliever and makes people feel more organized with what they are doing.

User can add a new note, and text can be containing anything the user wants to save, and multi-line is possible ([Figure 1.11](#)).

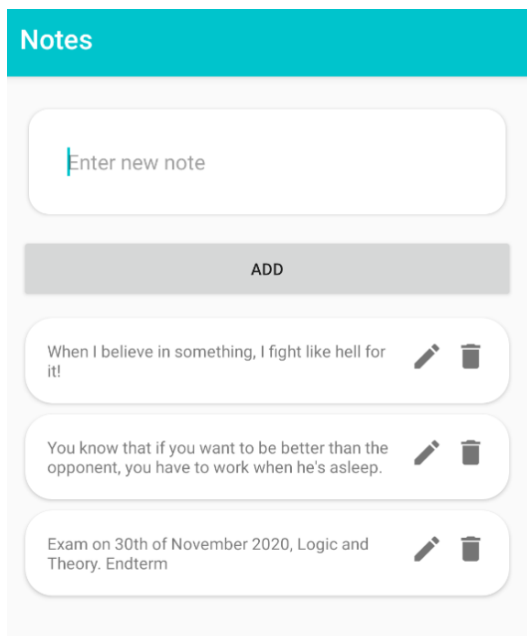


Figure 1.11 Notes Activity

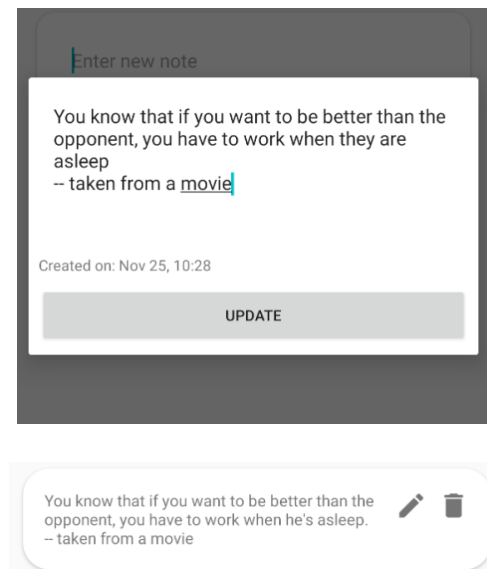


Figure 1.11.1 Update and delete notes

User can also edit note by clicking on the pen on the right-hand side of the listed note, we can see that after the note is added to the layout. Inside the editing note dialog, the user is able to see the created time of the note as well. Along with editing the note, we have the option to also delete the note from the layout. We can do so by clicking on the trash image, which is on the right side along with the pen image ([Figure 1.11.1](#)).

2.3.9 To do's

There are many things we have to do on daily basis but sometimes we forget to do them and therefore this application has the option of “TO DO”. This feature allows you to keep track of things you have to do on a certain day. This activity does not necessarily mean is only for medication. It can be for any kind of activity the user is planning to do or has to do. To-Do can be used to log the doctor's visit date and advice which came from the doctor. User can note on medication names as well next visit at the doctor or tests which the doctors have prescribed for the patient (user).

To-Do can be used for other purposes as well, like the vitamins or food you need to take before taking the medication or doing the tests. Nowadays, the dynamic of daily life is overloaded, and we forget about small things that are relevant to our health and wellbeing and using To-Do option as a reminder will help us organize ourselves better and consequently taking care of ourselves. Users can add a To-Do by clicking on the add button on the down right-hand side of the page ([Figure 1.12](#)). When the user types on add, another window will come up next with two fields, which need to be filled, with the subject and date when this activity will take place or happen ([Figure 1.12.1](#)). Users can only select dates in the future, past dates are disabled. After the user has chosen the subject and date, they will save the activity by clicking on the save button on the top right-hand side. After saving the activity, the user is redirected back to the rest of the saved activities. Considering the user has accomplished the task, on the right side of each to do activity is a radio button, the user can click on that, and that would mean that the task is completed.

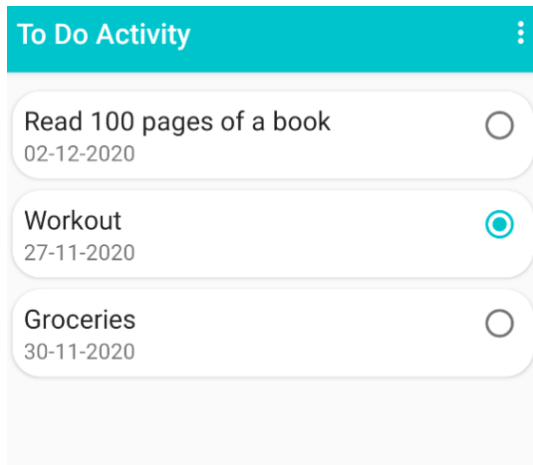


Figure 1.12 To-do activity

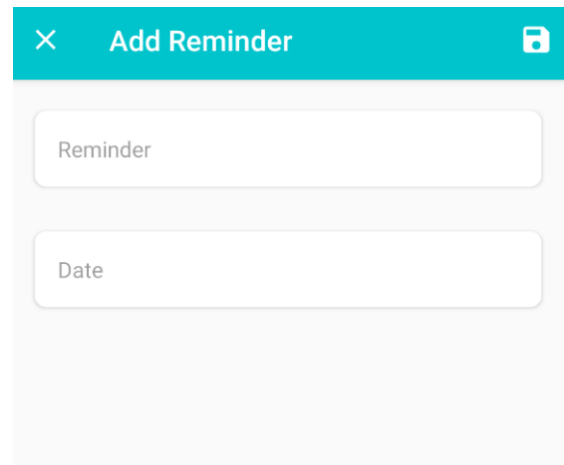


Figure 1.12.1 Add to-do activity

User can also delete the activity from the layout just by swiping right or left ([Figure 1.13](#)), or in the case when they want to delete all activities at once, they can do so by opening the drop-down menu on the top right-hand side, which has the option to delete all activities ([Figure 1.13.1](#)).

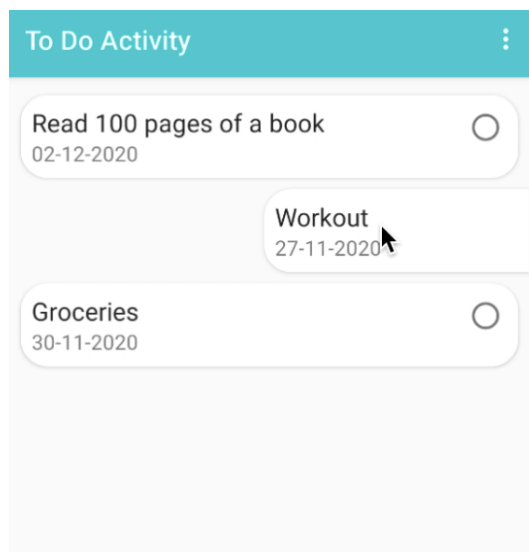


Figure 1.13 Swipe right or left to delete

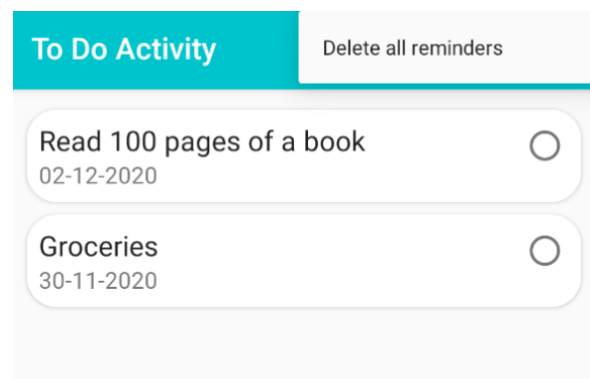


Figure 1.13.1 Menu option to delete all

2.3.10 Healthy tips

To stay healthy there are many preconditions which are to be followed by everyone. One of the most important issues is how much we care about what we eat. Eating healthy is one of the preconditions to stay healthy. As we get older, we need to be more careful about our health and

food we should avoid or is affecting our health. Normally, doctors always advise patients to eat healthily and not overload themselves with food, which contains e.g., sugar or fat etc.

To stay healthy, the human body needs sports activity and a healthy diet, which will enable the body to stay fit at all, times and avoid health issues. Healthy tips activity will offer the user some tips, in a layout of the image view ([Figure 1.14](#)). Users can swipe through the pictures and they will find the subjects containing the tips, which will be useful to the user's daily lifestyle.

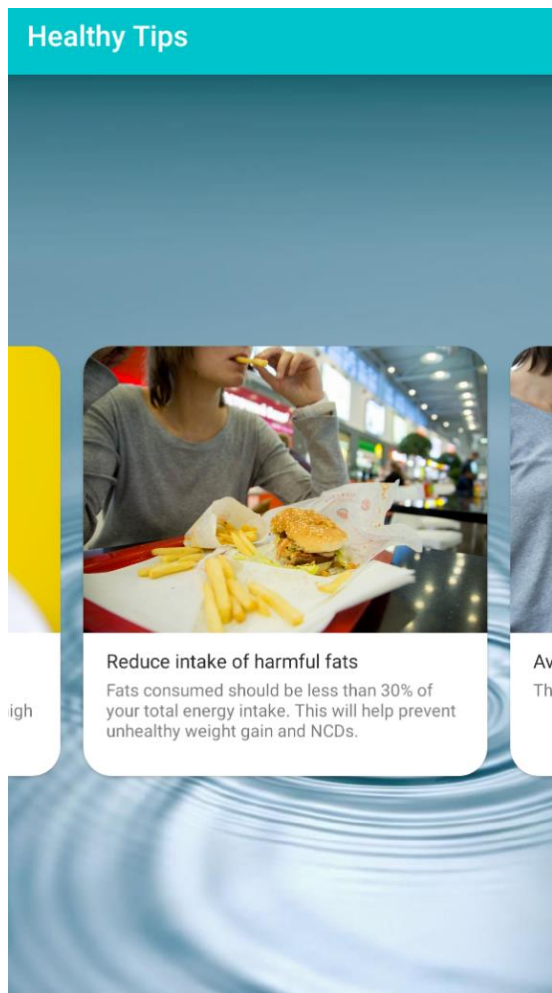


Figure 1.14 Healthy tips activity

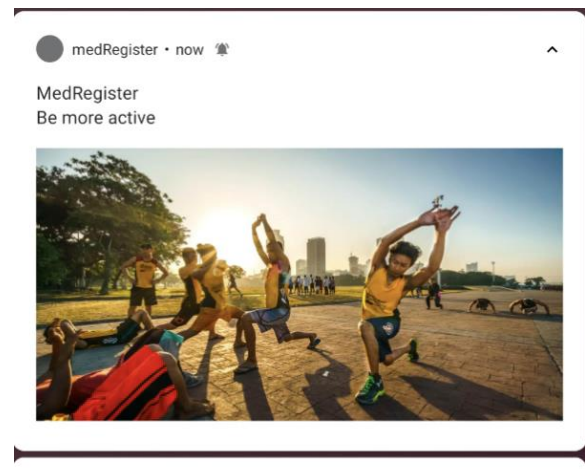


Figure 1.14.1 Daily notification

Additionally, to healthy tips from the application, users will also get daily notifications ([Figure 1.14.1](#)) with tips of the day, to motivate them into working harder or be careful of things they didn't know about.

2.3.11 Multi-language support

This application is to serve everyone, wherever you are and therefore it has multi-language support. Humans, wherever they are, have the same issues and the same topics that need to be addressed, therefore the application has different languages making life easy for people in different countries. When you choose the preface language, the whole application will change the setting on the chosen language by the user. All inside data will be operational in the language the user has selected to use. The currently supported languages other than English are Albanian ([Figure 1.15](#)) and Hungarian language ([Figure 1.15.1](#)).



Figure 1.15 Albanian language



Figure 1.15.1 Hungarian language

2.3.12 Dark mode

Most of the time, people use phones for different purposes and sometimes our eyes feel tired or blinded by the light of the screen. Technology has advanced and it was applied to this application, enabling the possibility of dark mode. With this setting, the user can change the white screen to black and therefore it will be no white reflection to the user's eyes. Knowing that most people use their phones for a considerable amount of time daily, it was necessary to set up this feature to help users feel more comfortable and have dual options ([Figure 1.16](#)).

When the device settings are changed to dark mode for the entire device, the application will change as well, so according to device settings, light mode and dark mode will both be available.

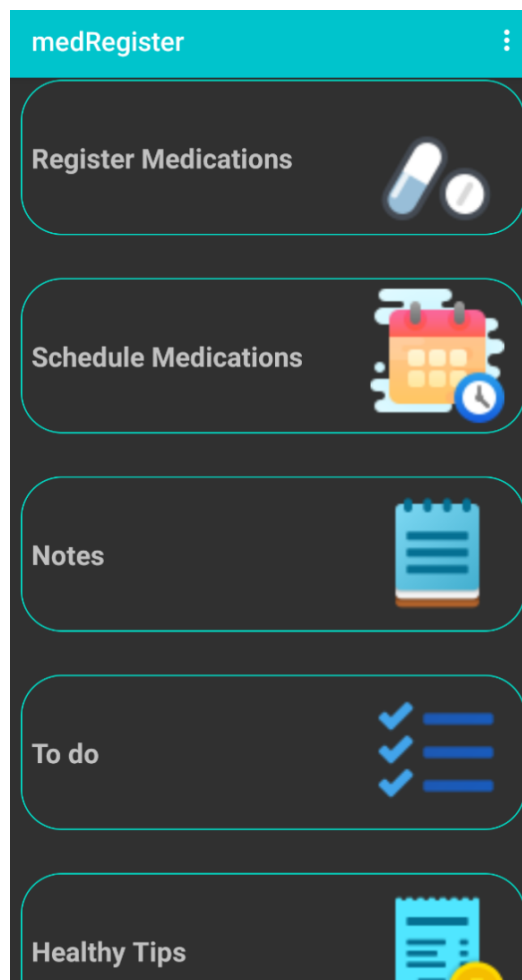


Figure 1.16 Dark mode

Chapter 3

Developer Documentation

This android application is created using a diverse set of tools and technologies. It goes through multiple stages, starting from developing to building and testing it through a pipeline, which is connected with GitHub, to build and test automatically on every merge/commit. I will explain everything about it in more detail later on in this chapter.

3.1 Creating the program

This application is written using Java as a programming language of choice, which is a very strong language for writing applications that can be used by single devices or even more divided systems. In order to be able to start building the application, an IDE is needed which is a very important part of the process of creating an application, it offers us so many tools to use just by the tip of our hands. Being able to use advanced IDE such as Android Studio, today saves a great amount of time and offers the chance at implementing multiple tools on an application considerably easy. To create this application Android Studio IDE [2] was used, which comes along with an Android Emulator [2] that makes it more convenient to test our code on multiple devices, multiple versions. All we need to do is create a virtual device, pick the type of phone we want to use, and the Android version that we want to come with it. There are certain system requirements needed in order to be able to use Android Studio and an Android Emulator, the same system requirements as discussed in the user documentation apply to developer documentation as well.

3.2 Program architecture

This android application contains certain technologies such as:

1. Firebase – backend system ([Section 3.5](#))
2. Activities – lifecycle ([Section 3.6](#))
3. MVVM Architecture ([Section 3.7](#))
4. Room databases – local database ([Section 3.8](#))

5. CI – Jenkins automation server ([Section 3.9](#))
6. Testing – contains UI and Unit testing ([Section 3.10](#))

3.3 Use-Case Diagram

One of the main purposes of this Android application is to be user friendly, simple to use and very straightforward to what you can use the application for. Graphical user interface is practical and very easily operated by anyone. The main goal is for even the elder generations to be able to use the application without having any issues while going through its functionalities. What users need to do first is register a valid account, that is important because the application has a verification process after registration. After logging in with the registered account, user will be forwarded to the main page which contains the main activities, such as registering medications and track their usage, schedule when to take a certain medication, writing notes and to do activities to keep their goals on track, and last but not least users will receive some healthy tips which will be useful to their lifestyle ([Figure 2.1](#)).

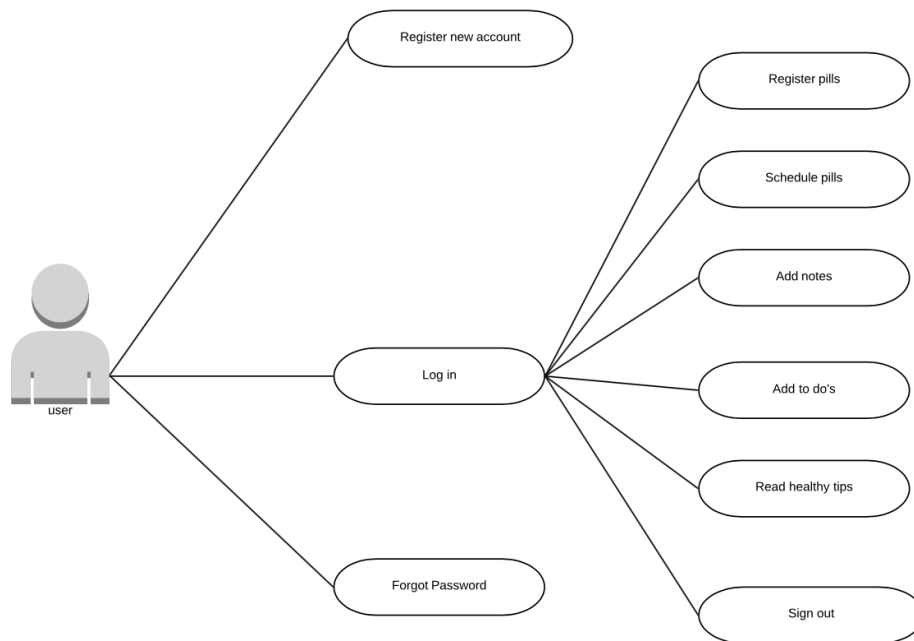


Figure 2.1 Use-case diagram

3.4 Program structure

The application has been created using different types of java classes, such as activities, adapters, data, dialogs and models ([Figure 2.2](#)) [2]. The main classes are the activities which are the most important classes in android development, adapters which makes the connection between the adapter and the data that is viewed. Every time we add items using the activities, the adapter listing will show us the data which is saved. We have databases classes which are used for storing the data. Dialog classes which pop up as alert windows when we click on a certain button, such as when user wants to sign out. Dialog window will ask the user if they really want to sign out, or stay logged in. And last model classes which hold the classes which are used to modify the information, we keep the setters, getters and constructors inside model class. More information about the importance of these classes will follow in the next sections.

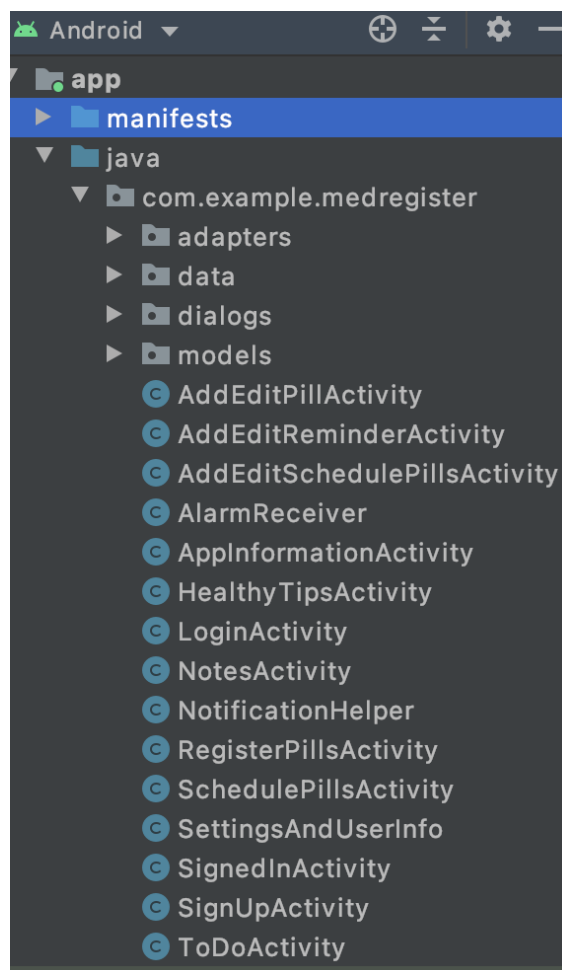


Figure 2.2 JAVA Classes

3.5 Firebase

Firebase [3] is a great backend platform which is operated by Google and can be used for android development. Firebase offers multiple tools to use, in order to make your android application more advanced, secure and easily scalable. In this project Firebase was used for user authentication, saving data in the real time database, cloud messaging for push notifications and crash analytics. As we can see from the image below ([Figure 2.3](#)), the client's applications using the SDKs provided by Firebase collaborate straight forward with the server [4].

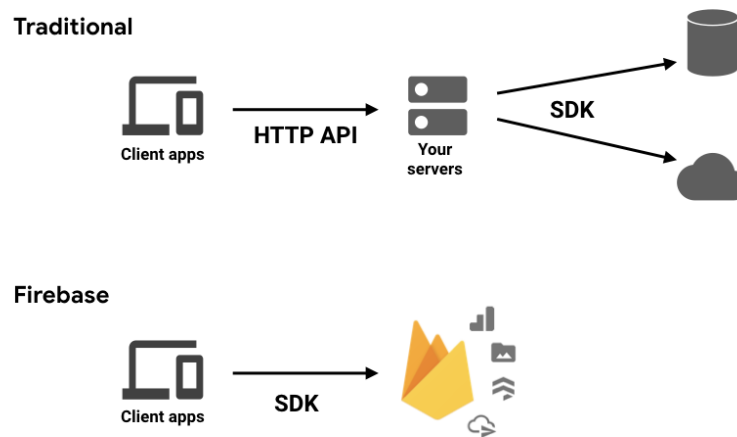


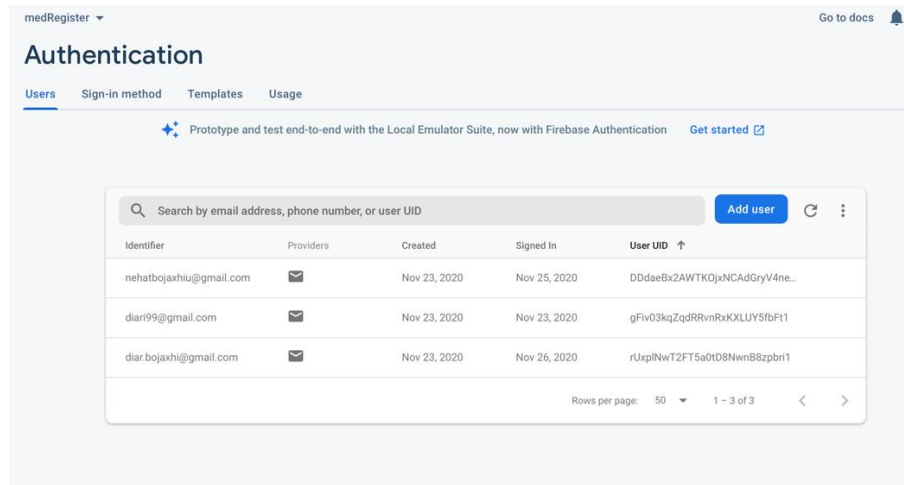
Figure 2.3 Firebase

3.5.1 Firebase User Authentication

Every program, we use every day, we are always afraid if our data is stored securely, or if someone can access our information which can be at times sensitive data. For this reason, I have used Firebase user authentication [3], which offers users a registration with a valid account, after registering every user receives verification URL, which in case they don't verify, their account won't be accessible. There are steps that go from Registering an account, to being able to log in inside the application. Firstly, in order to register user needs to use a complex password, which should contain at least one digit, one lowercase letter, uppercase letter, one special character and length of at least 8 characters and a maximum of 20. The checking of this password complexity is done using regular expression, which won't allow registration if the password doesn't match to the pattern required [5].

In order to add Firebase authentication to the android application we need to implement Firebase authentication libraries under dependencies in Gradle files.

After users are registered with the account, user information will show up in the console under authentication ([Figure 2.4](#)). From the console we can disable accounts, delete accounts, or reset password if user has any issues and isn't able to do so by itself. The only enabled sign in method provider is email.



The screenshot shows the Firebase Authentication console for a project named 'medRegister'. The 'Users' tab is selected, displaying a table of registered users. The table has columns for Identifier, Providers, Created, Signed In, and User UID. There are three users listed, all created on November 23, 2020. The first user is 'nehatbojaxhiu@gmail.com' with a signed-in date of November 25, 2020. The second user is 'diari99@gmail.com' with a signed-in date of November 23, 2020. The third user is 'diar.bojaxhi@gmail.com' with a signed-in date of November 26, 2020. At the bottom of the table, it indicates 'Rows per page: 50' and '1 - 3 of 3'.

Identifier	Providers	Created	Signed In	User UID
nehatbojaxhiu@gmail.com	📧	Nov 23, 2020	Nov 25, 2020	DDdaeBx2AWTKOjxNCadGryV4ne...
diari99@gmail.com	📧	Nov 23, 2020	Nov 23, 2020	gFiv03kqZqdRRvnRxKXLUYSfbF1
diar.bojaxhi@gmail.com	📧	Nov 23, 2020	Nov 26, 2020	rUxplNwT2FT5a0tD8NwnB8zpbri1

Figure 2.4 User authentication console

To authenticate users, we have to use a set of different activities such as, sign up with a new account activity, Log in activity and after we are logged in, we come to the main page activity. Every time user opens the application, we need to check if the user is authenticated, and if so then we stay logged in ([Figure 2.5](#)) [6].

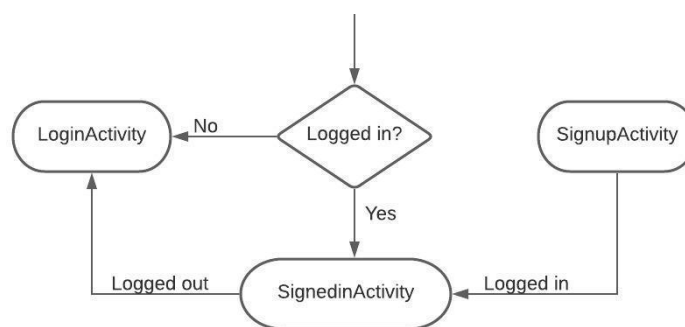


Figure 2.5 Authentication cycle

3.5.2 Firebase Realtime Database

When creating an application, database is one of the most important parts. Users need to store their data somewhere while using the application and be able to access the data in anytime. For this reason, we have Firebase Realtime Database which uses information simultaneously, every time the user makes a change, it will update within a very short amount of time.

Firebase Realtime Database is a NoSQL database which stores all the information in JSON format [3]. As one would assume, the main concern is security, how secure is firebase database. For that Firebase offers a rules language, which can be found in Firebase console under Security Rules, and from there we can decide how the data stored will be constructed and how it will be accessed by users. In this application I have used Firebase Database only in one activity, which is found in Account Settings. Firebase database is used there so that users can store their personal information, such as name or nickname and a phone number. In order to update the data after filling the fields, we have to click on the save button, and it will automatically update the data in real-time within milliseconds. This is how we see the data after it is saved from inside the application. Upon registration, `user_id` is saved to the database initially, for each registered user. For name and phone number, the user can write their own personal information, as well as update them at all times (Figure 2.6).

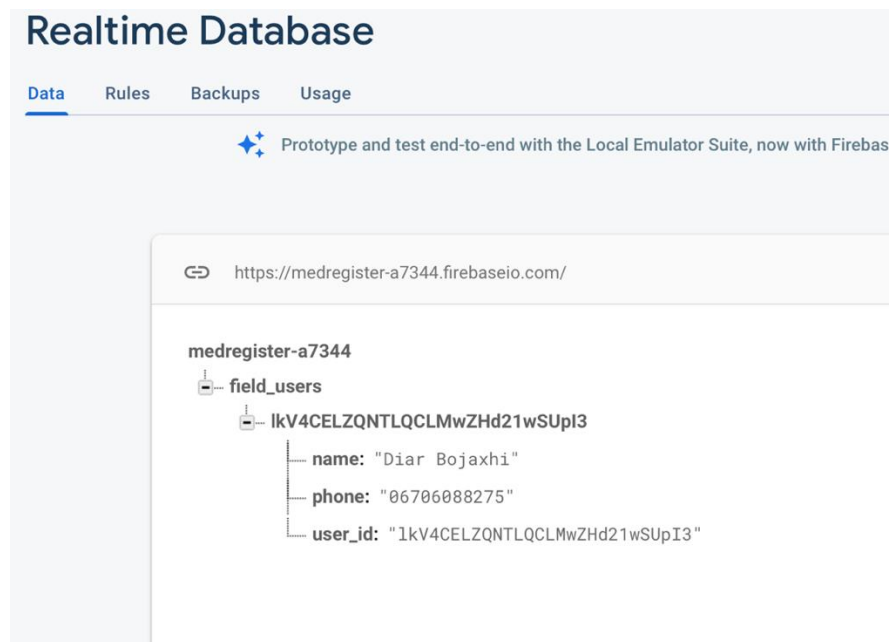
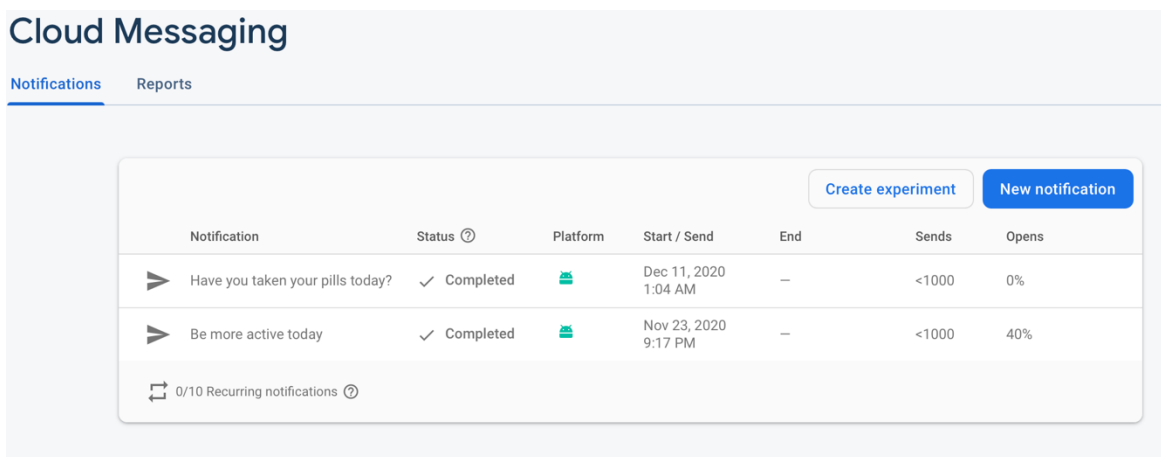


Figure 2.6 Realtime Database console

3.5.3 Firebase Cloud Messaging

Communication between the developer and its users can be very important in how successful an application can be. With firebase cloud messaging [3] developer can reach to its users by sending push notifications directly to all users at the same time. We can send push notifications to users in real-time directly from the console in just a couple of steps ([Figure 2.7](#)). There are multiple possibilities, such as we can schedule different notifications to be sent at all users at a particular time. We can also be more particular in case we want to target specific users, for example depending on their location, or their language. Another option is to target users which haven't been using the application for more than some days.



The screenshot shows the 'Cloud Messaging' interface with the 'Notifications' tab selected. It features a table with columns for Notification, Status, Platform, Start / Send, End, Sends, and Opens. Two notifications are listed: 'Have you taken your pills today?' and 'Be more active today'. Both are marked as 'Completed' and sent to Android. The first notification was sent on Dec 11, 2020, at 1:04 AM, with less than 1000 sends and 0% opens. The second was sent on Nov 23, 2020, at 9:17 PM, with less than 1000 sends and 40% opens. There are also buttons for 'Create experiment' and 'New notification'.

Notification	Status	Platform	Start / Send	End	Sends	Opens
▶ Have you taken your pills today?	✓ Completed	Android	Dec 11, 2020 1:04 AM	—	<1000	0%
▶ Be more active today	✓ Completed	Android	Nov 23, 2020 9:17 PM	—	<1000	40%

0/10 Recurring notifications

Figure 2.7 Cloud messaging, push notifications

3.5.4 Firebase Crashlytics

Firebase Crashlytics [3] is another tool which Firebase offers, that provides crash analytics. It's a real-time tool, mostly used to track performance issues of the application while it's in production. If user comes across a problem while using the application, and the application crashes, then the developer will be notified in the console of firebase with the issue that occurred to the user, details about the issue and recommendations on how it can be fixed. This way we can keep track of every issue that users run into while operating with the application.

3.6 Activities

Activity classes are one of the most important classes in android development. This is the class that provides the user interface to the application. Every activity represents a single screen. We have activities for each screen in the application, every time we switch from one page to the other, different activities communicate with each other. To switch from one activity to the other we need to use Intents. Every time we add new activities there are certain steps, we should take in order to fully implement an activity. First thing is we need to connect it with a layout file, which will define the appearance of the activity. Along with that, we need to modify Android Manifest file as well. Activities go through multiple states when the user navigates through the application, when user leaves the application and when comes back to the application ([Figure 2.8](#)) [2].

The states that the activities go through are called by itself, and there is no necessity for us to call these methods, unless however let's say we want to run a particular function every time the activity is on a specific state, then we can override the methods.

Whenever we click on any button that takes us to another activity, there are certain steps that happen. First method to be called is onCreate() method, which creates the layout that shows the view to the user. So after onCreate() which created the activity, then once the activity is created the onStart() method will be called, in this state the activity is started. Right after onStart() the next state that comes is onResume(). So all these methods onCreate(), onStart() and onResume() will be called automatically one by one once the activity is initiated.

After that if we want to leave the application and use our phone for something else, we push the home button which will take us out from the application. When the home button is pushed, the application will be minimized, which means the application is still in the memory. In order to keep it in the memory, onPause() method will be called first and then onStop(). And now that the application is minimized, if we go back to using it will call onRestart() method, which will again run onStart() and onResume() methods. In the case when we want to close the activity and go back, onPause(), onStop() and onDestroy() methods will be called and the activity will be destroyed.

The last case is, when the user is using other applications and they require some extra memory, then the operating system will close some applications. So in this event, when we go back to the application, onCreate() will be called once again from the start.

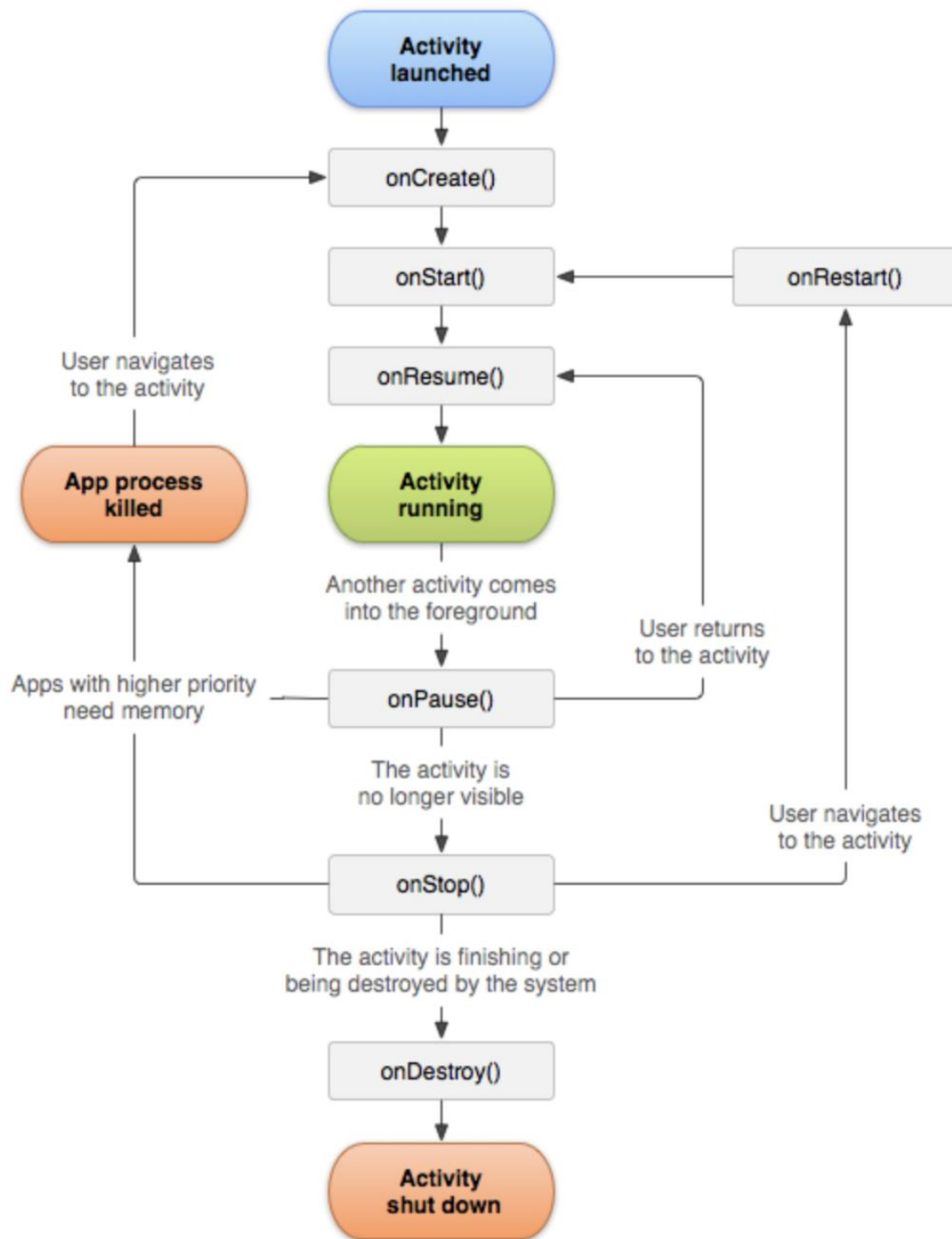


Figure 2.8 Activity lifecycle

3.7 MVVM Architecture

MVVM Architecture was used to build this application. MVVM means Model-View-ViewModel. This architecture is used in order to create an application in more separate modules, where each part of the program has a well-marked responsibility, and code can be modified in a simpler way, without having to change the structure ([Figure 2.9](#)) [2]. First thing we have is the model which indicates the class where we define the room entities. We have View which represents what the user sees when they open an activity. The connection between what the user can see and the logic behind that, is done by ViewModel. With the use of ViewModel we don't have to show the results to the user interface directly from the activity or fragment, we connect the activities or fragments with ViewModel which then gets all the data and draws it for the user to see in the screen. Between the ViewModel and the database object we have a repository class which gets the data from the database and makes it accessible to the ViewModel [7].

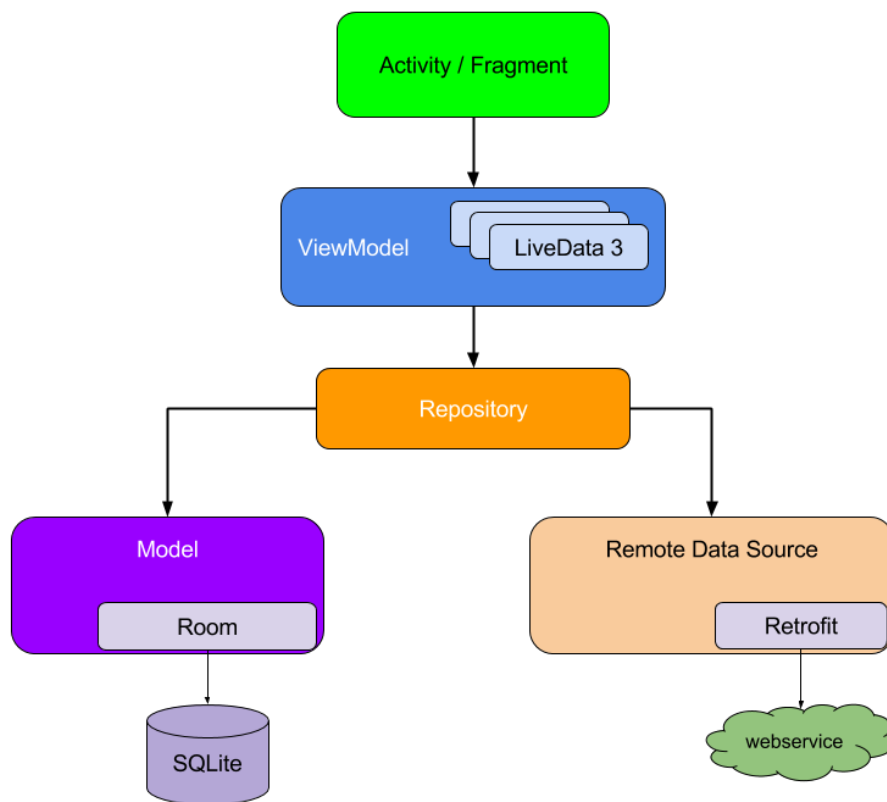


Figure 2.9 MVVM Architecture Diagram

Another architecture component used in this structure is LiveData which can be used to hold the data, and it can hold any type of data including lists [2]. LiveData can be observed by the UI controller, which means that every time the data changes the observer will be notified.

LiveData performs using the activity lifecycle, so according to the state that the application is in at the moment, LiveData will comply with it. In the image ([Figure 2.9](#)) we can see on the right-side remote data source, which is used when we need to fetch data from the internet. In this application this option wasn't used. We also have Room and Model, for which we will be explaining about in the next section.

3.8 Room Database

For users to be able to store data in the application, Room database was used, which is a local database [2]. Using Room has made it much easier, because we don't need to go through writing all the SQL queries which we need one by one. Instead, we call the methods, and Room will generate the SQL implementation. This way it makes it more convenient and safer, considering one small mistake in the database could turn into the application crashing, and if that would happen while in production it would be an issue. Another reason to use Room is because if there is something wrong, say the column which we are trying to query through doesn't exist, then the code won't even compile, which is much better that the issue is known before going to production. There are three parts we need to implement in order to have a fully functional Room databases. We need an abstract class that extends RoomDatabase and here we will build the database, we can also create a callback with pre-populated data, which is very recommended to test if the database is working correctly from the start. The second part we need is a model class, where we need to add Entity annotation, where we can specify the table name or in the case that we don't then table name will be automatically set to the name of the model class. And last, we have DAO which holds the methods that we need in order to retrieve the databases.

3.9 CI - Jenkins

CI or Continuous Integration has become a very important part on developing a software in a more efficient and reliable way [8]. Especially when it comes to bigger groups or organizations, and there are multiple people working on the same project. The aim of CI is to set up an approach to build and test applications automatically, on every small change. To develop this project, GitHub was used as the version control system using Git. Development was done using branching strategy, once a functionality was completed then merge the working branch into the master branch. After successful merging, the automated system will run the build and test jobs automatically and will display results if all the checks have passed successfully or there was an issue. The automated server that does the builds and testing, checking if everything was correct, is Jenkins ([Figure 2.10](#)), which is a free and open-source server [9]. In order to set up Jenkins server, I opted with Google Cloud Platform, as it allows 12 month of free use, and ~ 10 GB Debian Linux VM [10]. Jenkins URL is: <http://35.205.59.149:8080>, and in order to make it accessible with the 8080 port it was required to configure the settings in Google cloud platform console under the VM instances.

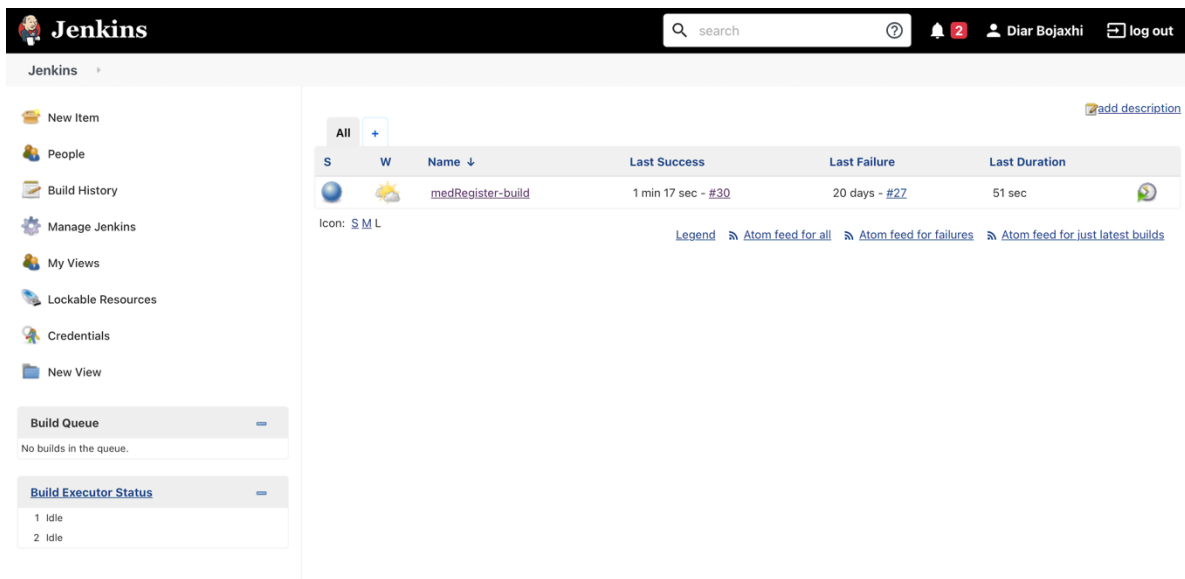


Figure 2.10 Jenkins Main Page

3.9.1 Connection Jenkins and GitHub

To connect Jenkins server with the GitHub project, we need to create a webhook in GitHub, where we need to specify the Jenkins URL. Afterwards we need to edit the configurations in Jenkins server, and that has to be done under Configure System where we need to make the connection with GitHub API. The rest of the configurations required are done inside the build-job ([Figure 2.11](#)). We need to check on the box that contains the hook trigger for GITScm polling.

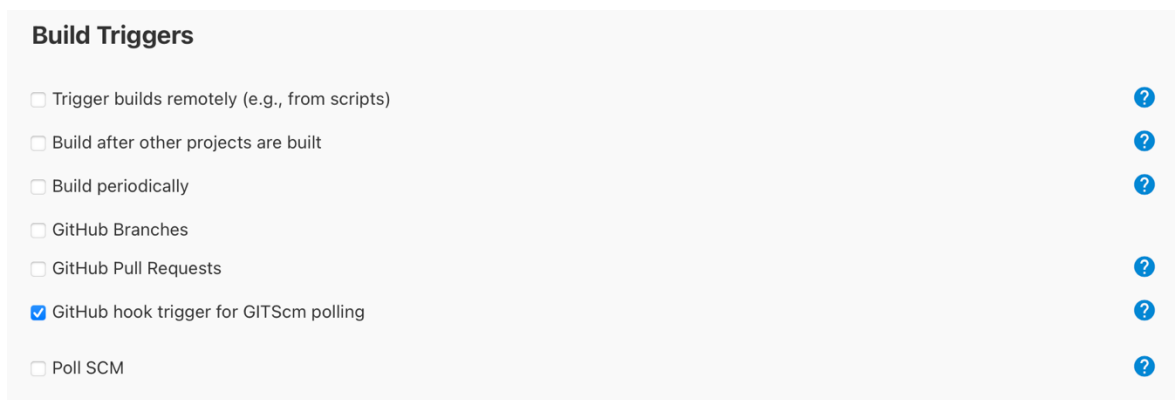


Figure 2.11 GitHub hook trigger

The next thing required is we need to set the source code management, and that will be the GitHub repository URL [1], and there we can also choose which branch we want the job to run for.

3.9.2 Build Job

Build job is used to build and test the application on every commit/merge automatically from GitHub ([Figure 2.12](#)). In order to build the android application, there are certain configurations required to be done on the Debian VM that is used for Jenkins. We need to install Java JDK, Android SDK tools which are very important in order to test with an emulator and accept the Android SDK license [11]. After all, we need to do is create a new job, and add build actions such as clean, build, test. And as a post-build task, I have enabled an extra tool such as Lint checker [12]. After every build Lint checker will give recommendations for the changes that could be done, in order to have a more efficient and better performance application.

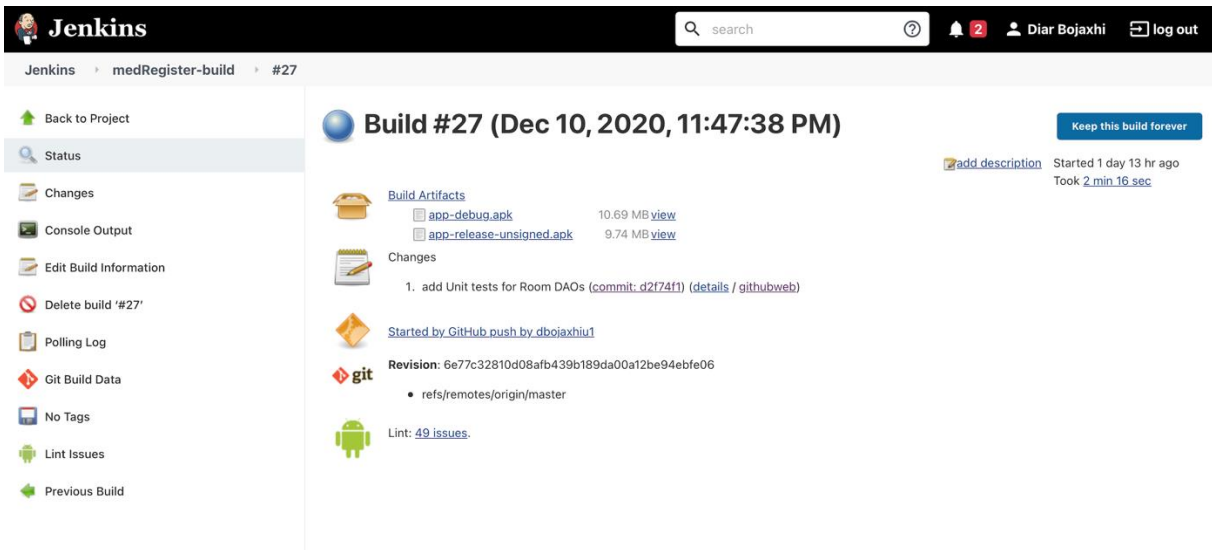


Figure 2.12 Build status

As another post-build action that has been added is setting the GitHub commit status, according to the job execution result, if the checks have passed or not (Figure 2.13). In order to set up this connection, a personal access token has to be generated in GitHub for the project, and then we can use it in Jenkins to finalize the connection.

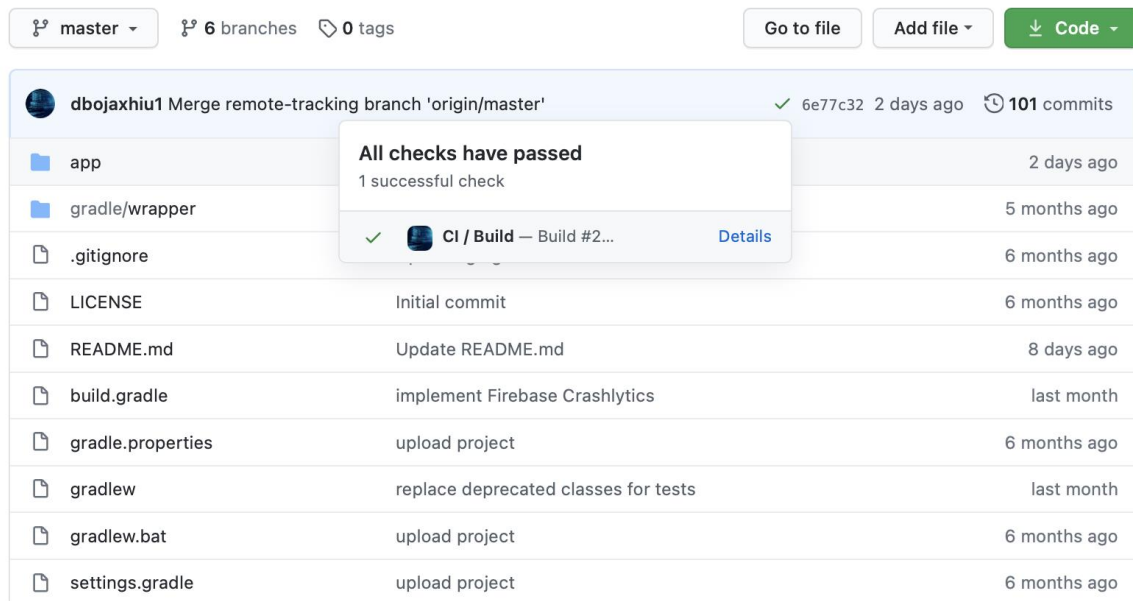
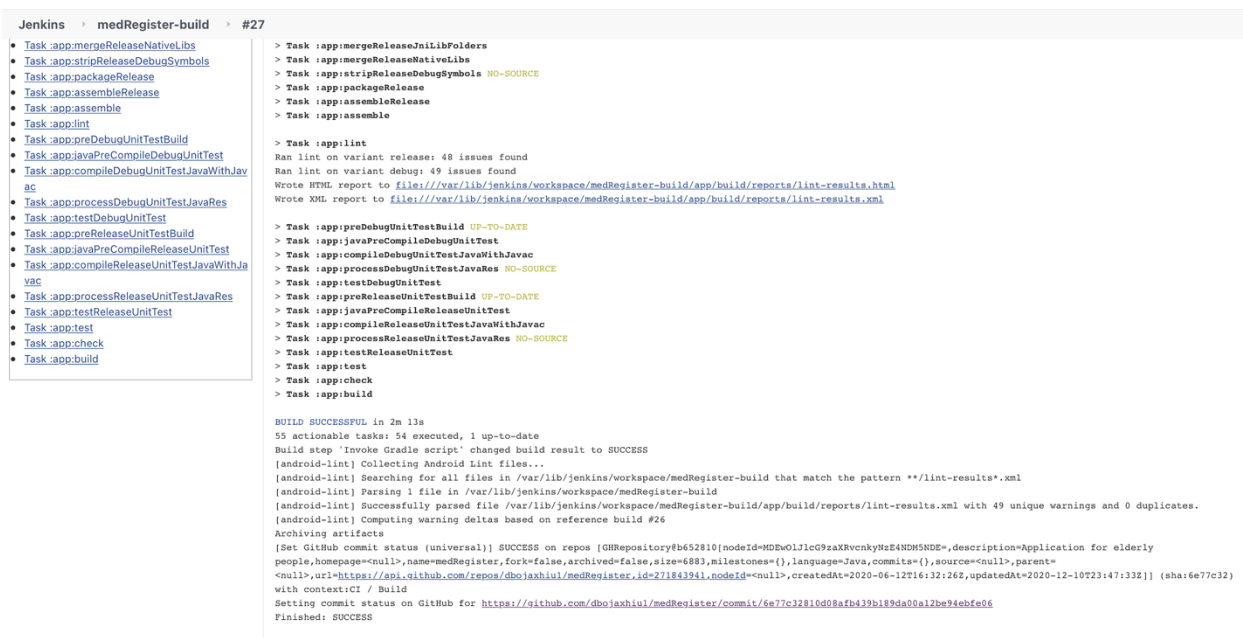


Figure 2.13 Build status on GitHub repository

3.10 Testing

Testing is another very important part in developing a successful software. This way we can find any issues that we might have but couldn't notice while developing. To test this application, I have written UI tests and unit tests using JUnit and Espresso testing framework [2]. After writing tests with Espresso [13], we are able to see the association between different activities and functionalities in the application. These tests will run performing automatic clicks and moves according to the test code, in a much faster way, and it will use the application in a way that will test possible movements. That could be also done by manual testing, as if we give the application to another person to test it, but it wouldn't be as efficient and safe. We can execute tests from inside the application using an Android Emulator. Another way to check if the tests have been executed successfully is from the console output that comes from running the job in Jenkins (Figure 2.14).



```
Jenkins > medRegister-build > #27
• Task :app:mergeReleaseNativeLibs
• Task :app:stripReleaseDebugSymbols
• Task :app:packageRelease
• Task :app:assembleRelease
• Task :app:assemble
• Task :app:lint
• Task :app:preDebugUnitTestBuild
• Task :app:javaPreCompileDebugUnitTest
• Task :app:compileDebugUnitTestJavaWithJavac
• Task :app:processDebugUnitTestJavaRes
• Task :app:testDebugUnitTest
• Task :app:preReleaseUnitTestBuild
• Task :app:javaPreCompileReleaseUnitTest
• Task :app:compileReleaseUnitTestJavaWithJavac
• Task :app:processReleaseUnitTestJavaRes
• Task :app:testReleaseUnitTest
• Task :app:test
• Task :app:check
• Task :app:build

> Task :app:mergeReleaseNativeLibs
> Task :app:stripReleaseDebugSymbols NO-SOURCE
> Task :app:packageRelease
> Task :app:assembleRelease
> Task :app:assemble

> Task :app:lint
Ran lint on variant release: 40 issues found
Ran lint on variant debug: 49 issues found
Wrote HTML report to file:///var/lib/jenkins/workspace/medRegister-build/app/build/reports/lint-results.html
Wrote XML report to file:///var/lib/jenkins/workspace/medRegister-build/app/build/reports/lint-results.xml

> Task :app:preDebugUnitTestBuild UP-TO-DATE
> Task :app:javaPreCompileDebugUnitTest
> Task :app:compileDebugUnitTestJavaWithJavac
> Task :app:processDebugUnitTestJavaRes NO-SOURCE
> Task :app:testDebugUnitTest
> Task :app:preReleaseUnitTestBuild UP-TO-DATE
> Task :app:javaPreCompileReleaseUnitTest
> Task :app:compileReleaseUnitTestJavaWithJavac
> Task :app:processReleaseUnitTestJavaRes NO-SOURCE
> Task :app:testReleaseUnitTest
> Task :app:test
> Task :app:check
> Task :app:build

BUILD SUCCESSFUL in 2m 13s
55 actionable tasks: 54 executed, 1 up-to-date
Build step 'Invoke Gradle script' changed build result to SUCCESS
[android-lint] Collecting Android Lint files...
[android-lint] Searching for all files in /var/lib/jenkins/workspace/medRegister-build that match the pattern **/lint-results*.xml
[android-lint] Parsing 1 file in /var/lib/jenkins/workspace/medRegister-build
[android-lint] Successfully parsed file /var/lib/jenkins/workspace/medRegister-build/app/build/reports/lint-results.xml with 49 unique warnings and 0 duplicates.
[android-lint] Computing warning deltas based on reference build #26
Archiving artifacts
[Set GitHub commit status (universal)] SUCCESS on repos [GitHubRepository#b652810[nodeId=HDEwO13cG9zaXKvcmkyZzE4NDM5NDc=,description=Application for elderly people,homepage=null],name=medRegister,fork=false,archived=false,size=6883,milestones={},language=Java,commits={},source=null,parent=null],url=https://api.github.com/repos/dbojxhiul/medRegister,id=271843941,nodeId=null,createdAt=2020-06-12T16:32:26Z,updatedAt=2020-12-10T23:47:33Z] (sha1:6e77c32)
with context:CI / Build
Setting commit status on GitHub for https://github.com/dbojxhiul/medRegister/commit/6e77c32810d08afb439b189da0a12be94ebfe06
Finished: SUCCESS
```

Figure 2.14 Jenkins job console output

3.10.1 UI and Unit Tests

To test this application UI tests [2] were written to check for the cases when we put some text on a EditText field, or when we click on a button. This way we can test user interactions and find out if there is any issue when user is performing different activities inside the application. Tests have been made to test activities and databases. For testing activities, there are certain steps to take, first thing we need to do is create a method setUp() which will help to get the activity. And after we got the activity we can go ahead and write the test methods. After we are done with the tests, the next method that comes after is, tearDown() method which will help in closing the activity. The annotation @Before is required to get the method executed firstly.

```
@Before
    public void setUp() throws Exception {
        todoActivity = todoActivityActivityTestRule.getActivity();
    }
```

And similar to setUp(), in tearDown() we also need to set the annotation, but in this case its different because we need this method to run last, after all the other test methods. Annotation in this case will be @After.

```
@After
    public void tearDown() throws Exception {
        todoActivity = null;
    }
```

Along with activities, databases have also been tested with unit tests. The process of testing goes very similar to activities, but here we firstly need to build the databases before running the tests. In this application LiveData has been used to hold the data, so in order to create the tests, we can't just treat it as a simple list, we need a LiveDataUtilTest class which was taken from [14] under Google LLC Copyrights. With the help of this class, I was able to test inserting data into a list, updating it and deleting.

@Test

```
public void insertPill() throws InterruptedException {
    Pill pill = new Pill("name", "instruction", 1, 20);
    database.pillDao().insert(pill);
    LiveDataUtilTest<List<Pill>> listLiveDataUtilTest = new
LiveDataUtilTest<>();
    List<Pill> insertedPills =
listLiveDataUtilTest.getOrAwaitValue(pillDao.getAllPills());

    assertNotNull(insertedPills);
    assertEquals(pill.getName(), insertedPills.get(0).getName());
    assertEquals(pill.getInstruction(), insertedPills.get(0).getInstruction());
    assertEquals(pill.getUsage(), insertedPills.get(0).getUsage());
    assertEquals(pill.getPackageContains(),
insertedPills.get(0).getPackageContains());
}
```

Chapter 4

Conclusion

This paper focuses on creating a more advance way of keeping track of medications, and other features such as scheduling activities when we need to take a certain medication. Other than for medication this application can be used for daily basis on writing notes and trying to keep up with goals by writing them down and setting a date to them, to help in reaching that goal. It can be concluded that having such an application into our lives can help people organize their daily routine and consequently take care of their health.

4.1 Future work

After completing this android application, there are a lot of opportunities and features we could add in order to make it more effective in tracking medications. There are a lot of different medications that people use every day and making the application extendable enough to be able to save all types of medications would be one of the main features that could be added. Another feature which would be useful to the user would be if users would be given reports of the last few months, containing data from the medications that the user used.

Bibliography

- [1] GitHub Public repository
URL: <https://github.com/dbojaxhiu1/medRegister>
- [2] Android Documentation. December 11, 2020.
URL: <https://developer.android.com>
- [3] Firebase Documentation. October 20, 2020.
URL: <https://firebase.google.com/docs>
- [4] What is Firebase? The complete story, abridged. November 15, 2020.
URL: <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>
- [5] Java lang String matches for regex. November 3, 2020.
URL: <https://www.geeksforgeeks.org/java-lang-string-matches-java/>
- [6] Firebase Login and Authentication. September 12, 2020.
URL: <https://blog.mindorks.com/firebase-login-and-authentication-android-tutorial>
- [7] MVVM Architecture. October 2, 2020.
URL: <https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1>
- [8] Continuous Integration. June 15, 2020.
URL: https://en.wikipedia.org/wiki/Continuous_integration
- [9] Jenkins Documentation. July 15, 2020.
URL: <https://www.jenkins.io/doc/>
- [10] Install Jenkins on Google Cloud Platform. June 20, 2020
URL: https://www.youtube.com/watch?v=l7ngjJ_RVXs
- [11] Preparing the environment for building an Android app, June 15, 2020
URL: <https://wiki.debian.org/AndroidTools/IntroBuildingApps>
- [12] Lint
URL: [https://en.wikipedia.org/wiki/Lint_\(software\)](https://en.wikipedia.org/wiki/Lint_(software))
- [13] Discovering espresso for Android. November 20, 2020.
URL: <https://testyour.app/blog/espresso-swiping>

- [14] Unit-testing LiveData and other common observability problems. November 25, 2020.
URL: <https://medium.com/androiddevelopers/unit-testing-livedata-and-other-common-observability-problems-bb477262eb04>
- [15] Android Documentation. December 11, 2020.
URL: <https://developer.android.com/guide>
- [16] Overview of Android Operating system. July 12, 2020.
URL: <https://web.stanford.edu/class/cs231m/lectures/lecture-2-android-dev.pdf>
- [17] Swipe to delete. October 10, 2020.
URL: <https://medium.com/@zackcosborn/step-by-step-recyclerview-swipe-to-delete-and-undo-7bbae1fce27e>
- [18] MaterialDateTimePicker. October 23, 2020.
URL: <https://github.com/wdullaer/MaterialDateTimePicker>
- [19] Model View View-Model (MVVM). October 18, 2020.
URL: <https://www.youtube.com/watch?v=ijXjCtCXcN4>
- [20] Setting up Jenkins on MacOSX, June 20, 2020.
URL: <https://medium.com/@ved.pandey/setting-up-jenkins-on-mac-osx-50d8fe16df9f>
- [21] Android ListView with ListAdapter. October 23, 2020.
<https://javatutorial.net/android-listview-with-listadapter-example>
- [22] Alarm Manager. November 3, 2020.
URL: <https://codinginflow.com/tutorials/android/alarmmanager>