
- Description
- Intended User
- Features
- User Interface Mocks
  - Screen 1
  - Screen 2
  - Screen 3
- Key Considerations
    - How will your app handle data persistence?
    - Describe any corner cases in the UX.
    - Describe any libraries you'll be using and share your reasoning for including them.
    - Describe how you will implement Google Play Services.
- Next Steps: Required Tasks
  - Task 1: Project Setup
  - Task 2: Implement UI for Each Activity and Fragment
  - Task 3: Create Widget
  - Task 4: Notifications
  - Task 5: Testing and Validation

GitHub Username: nathanafacion

Remedy Me

Description

The main goal of this app is to help people to have a better health by reminding when to use their medicine. We all know how hard it is to remind of all the times you need to take your medicines, especially when said medicine has strict times to be taken, and even worse, when you need to take more than one at a time. So, through the use of our app Remedy Me, we want to make this process easier, reminding you of these times.

Intended User

Every person that need to take a medicine.

Features

List the main features of your app. For example:

- List every medicine you need to take, with their defined time.
- Allow the creation of a new reminder
- Allow editing of already created reminders.
- Allow the removal of any reminder.
- Send notifications when the time has come to take any of the medicines
- Allow the configuration of an alarm clock for each medicine.

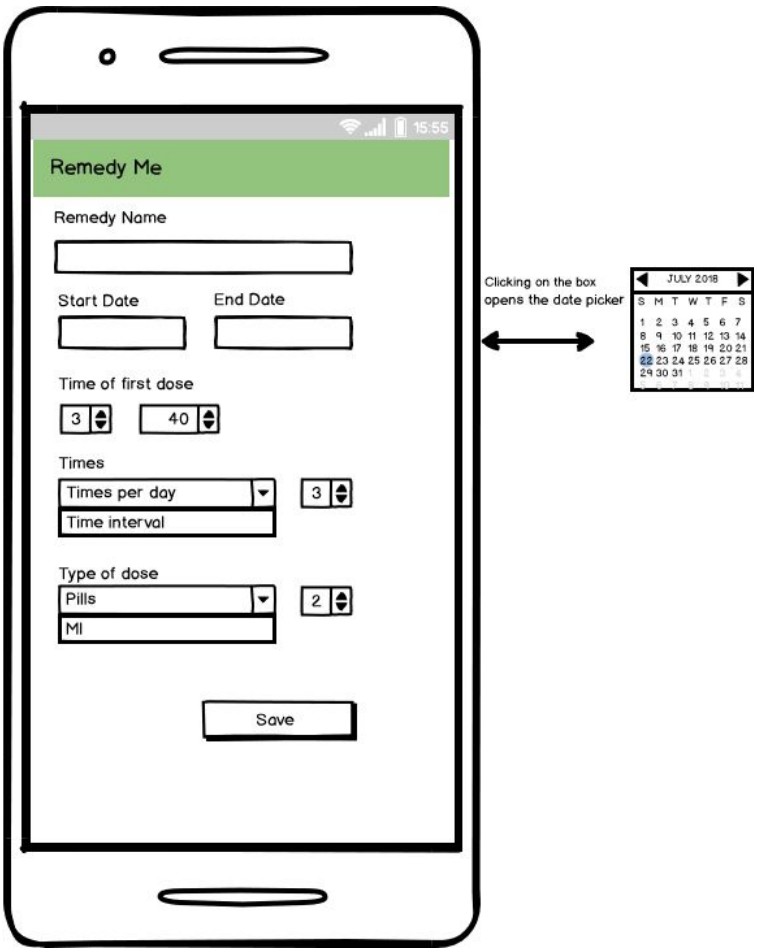
User Interface Mocks

Main Activity Screen

Main activity screen, shows all the Remedy reminders created by the user, ordered by time, where the first item is the next remedy to be taken. Each row shows the name of the remedy, the time of the next dose and the amount to be taken. The arrow button shows the next times the remedy must be taken, the pencil button calls the “New Remedy Activity”, with the

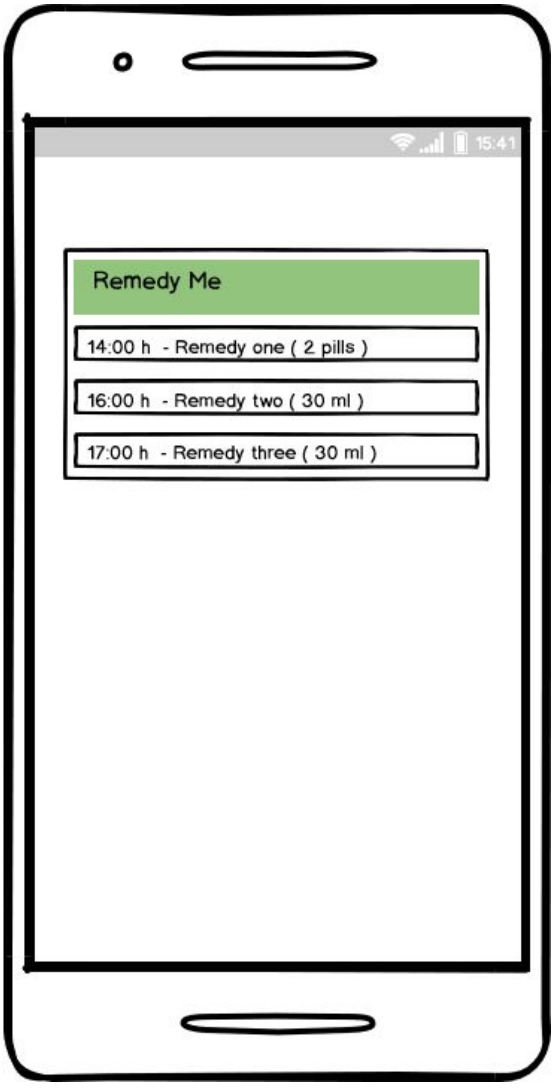
information of this row to be edited, and the trash button removes this Remedy reminder. There is also a FAB that calls the “New Remedy Activity” to create a new Remedy reminder. On the toolbar, there is a menu containing the backup option to send the local list to the Firebase Database.

New Remedy Activity



The New Remedy Activity is called when the user clicks the FAB on the Main Activity, or clicks on the pencil button on a Remedy reminder item of the list. This activity have all the needed information that have to be filled in order to create a new Reminder, or edit an existing one. By clicking on the Save button, the new Reminder is added to the local database, and the user is taken to the Main Activity Screen, where the new Remedy reminder will be shown.

Widget



The Widget will be a simplified version of the Main Activity screen, showing the times of the next remedies to be taken.

Key Considerations

How will your app handle data persistence?

The app will save the reminders created both locally, using SQLite database, and remotely using Google Firebase Database. The app will retrieve and send remote data using an IntentService.

Describe any edge or corner cases in the UX.

Our UX will use the Material Design Guidelines. The main screen with be a list with the created reminders, and a button that allow the creation of new reminders. There will be an option for editing and exclusion for every reminder.
If the list is empty, a textbox will appear at the top of the page, asking the user to create a reminder using the FAB button.
If there is no internet connection, and the user select the “Backup” option, a toast message will appear warning that this option is only available with a connection.
If the user tries to Login using it’s Google Account, but there is no internet, then the user will still be redirected to the list Activity, but a message will be shown, warning that because there was no connection, the login could not be made, and the data shown is the one stored locally.
Every string used in the project will be stored in the *strings.xml* for maintainability.
The app will also allow for the use of RTL on every screen.
And it will support accessibility through content descriptions.

Describe any libraries you’ll be using and share your reasoning for including them.

The app will be developed using only the Java language, version 9.

We will be using the SQLite library, version 3.23.1, to store data locally on the device, as it is the Android standard. This will be done using a Content Provider and Loaders, storing the list of reminders created by the user, storing its name, time, and other settings.

We will be using the Google Firebase library, version 15.0.0, to store data remotely, as it is a reliable and well known set of databases.

We will be using the Google login library, version 15.0.0, to sync the user’s data with the remote database. And we will be using the Material Design library, to make our app’s interface clean, intuitive and beautiful.

We will use the Google AdMob library, version 15.0.0, to display a small ad banner in the bottom of the Main Activity screen.

We will be using the Butterknife library, version 8.8.1, to bind views in the code in a easier way than the standard Android method, and save some development time.

Finally, the app will be developed using the gradle version 4.8.1 and Android Studio 3.1.3.

These versions might change if compatibility problems arise during development, but I intend to use the latest version possible.

Describe how you will implement Google Play Services or other external services.

We will be using the Google Login service to identify a user on the device, and we will be using the Google Firebase Database to store remote data. When logged in with it's Google account, we will allow the user to save the reminder list remotely. This will be done via a "Backup" button, that will send all information needed to the Firebase Database. There will be a button to Sync the local to list with the remote list. This way, the user can have its medicine list in a different device.

Next Steps: Required Tasks

Task 1: Project Setup

Setup the needed databases and libraries.

- Configure libraries
- Configure the Firebase Database
- Configure the SQLite local Database

Task 2: Implement UI for Each Activity and Fragment

Build the UI for the Main Activity, that will have the Reminder List, a FAB to create a new reminder, which will call the Create Reminder Activity, a toolbar and toolbar button that will contain the "Backup" button, that stores the local database to the Firebase Database, and a "Sync" button, that download the data stored in the Firebase to the local SQLite database..
Build the UI for the list item fragment, with the needed information, and a button for edition and deletion.
Build the UI for the Create Reminder Activity that will have the following fields:
A Text field for the Medicine name;
A Calendar with the starting and ending dates for the medicine;
A Text field for the starting time of the first medicine intake (i.e. at 8:00), which will be used to set the times of the other uses, based on the timing option.
A combobox with options for the timing of the medicine. The first option is to set the interval the medicine must be taken (i.e. every 8 hours). The second is to set how many times a day it must be taken (i.e. 4 times a day).

A second combobox with the type of medicine to be taken, having the option to be a “Pill” for pill type medicine, or a “ml” for liquid based medicine (i.e. a cough syrup), followed by a text field for the dosage.

Task 3: Create Widget

Build the widget UI, that will show the 3 next medicines to be taken, with their time and dosage.

Task 4: Notifications

Create the notifications, that will warn the user of the Medicine name, the time and dosage.
These will be configurable with an associated alarm clock, having the option to be disabled.

Task 5: Testing and Validation

Validate for connection failures, and guarantee the application does not crash when there is no internet connection.
Validate if the user can access it’s local list even without an internet connection.
Validate if the user can access the app even if there is a problem with the google login, and access it’s local data.
There will be no Unit test for these validations.