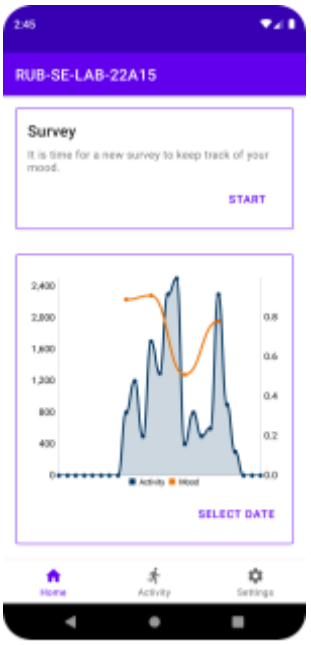
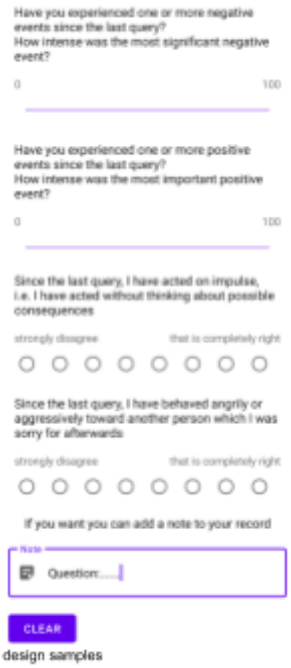
Final- Report

Application overview



Home screen



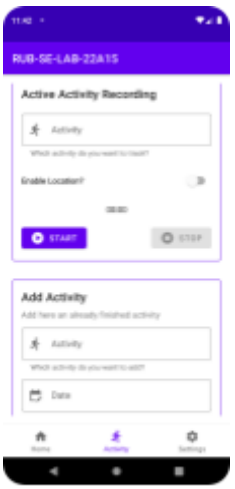
design samples

For the purpose of this course, we were asked to develop an application for our customer Mr Professor Markus Reichert of the sports faculty at Ruhr-University Bochum. Professor Reichert's research interest is to find a relation between human motions/exercises and mental health. For his research, he needs datasets of motions and mood surveys. The easiest way to generate a large number of datasets in our digitized life is to use a modern mobile phone; the smartphone, which we can assume that the majority of mankind in developed countries have. A smartphone contains different sensors to detect the motion of the device. The accelerometer is one of the basic sensors besides the Global-Positioning-System sensor for localization; which are suitable for the application to generate raw datasets for the researchers. Besides the technical requirements, there are also logical, security and privacy requirements that the electronic health application should meet.

The smart app has a great variety of functionalities, which are going to be clearly presented in the following lines. The primary aim of this app is to give a user the possibility to record his or her acceleration via sensor data and mood through a survey which he or she should provide the data to Professor Reichert's research team. The user has the possibility to start a survey simply by clicking on the start button located on the home screen (welcome screen) which will lead him/her to the various questions regarding the instant mood. The user can decide to interrupt the survey when he/she wishes and if a mistake was made while answering the questions, the user has the possibility to return to the previous question without changing the already given answers. Furthermore, the user will be asked to give a reason why he or she wishes to interrupt the survey. At the end of the questionnaire, the user can either add or not a note to the survey before saving the answers to the questions by clicking the save button. The survey can be taken repeatedly as long as the user desires or better still, the user can set a notification on when to be reminded to take the survey. Therefore the users can find his/her development

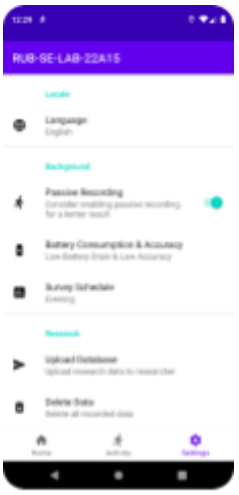
process in a chart on the home screen as well. The application stores a deviation of the researcher's database for the presentation on the home screen. If the user wants to see their progress in a fixed time interval; they can select the desired date for showing this. The acceleration, GPS and answers to the questions will be saved in the mobile phone's SQLite local database using Room [L1]. Room is a persistence library and provides an abstraction layer for the underlying SQLite database. It has three advantages:

- "Compile-time verification of SQL queries.
- Convenience annotations that minimize repetitive and error-prone boilerplate code.
- Streamlined database migration paths." [L2]



activity recording

From time to time, the user can upload the locally stored database files with the upload button on the setting fragment to Google's Firebase [L4] cloud storage solution. Firebase pledges robust operations and strong security. Thanks to the combination of the local database and cloud storage, this application does not need a server for data exchange. The key point: within Firebase, it is possible to generate a user-id without an email address or name, and this specific user-id helps to identify the recording devices, which ensures the pseudonymity and anonymity of the user except when the users log their activities. The researchers can access and download the database files on Firebase easily. After the download; the entities of the database file can be shown by using the "DB Viewer for SQLite". Further supporting services were implemented so that the data saving continues when the user switches to another application or turns it off.



Settings

Another key functionality of the App is the opportunity to switch on the passive acceleration recording in the settings. This "turn on and forget" feature can be used as an additional daily measurement. There are three options to set up acceleration sensor frequencies which help to reduce the power consumption of the passive recording. The application also has the possibility to save the preferred day time to do the survey.

Another major functionality of this app is the fact that the user can record his/her location all day long as long as the Location is enabled on the next window(activity session) as you can also see on the image below. When the activity recording is started, a timer will also be started to remember the user if he/she has forgotten to stop the recording after 3 hours. But before enabling the Location, you need to choose how you wish to enable it(image below). In the particular session of the app, the user can start an activity at a given time and on a certain day, in order to easily get access to the data days after. Once the user gives in details of his/her wished

activity, the user can view an activity icon on the menu-bar at the top of the page indicating that the activity is being recorded(view picture below) and could be stopped after clicking the stop button. After clicking on the stop button, the user can either save or discard the recording.

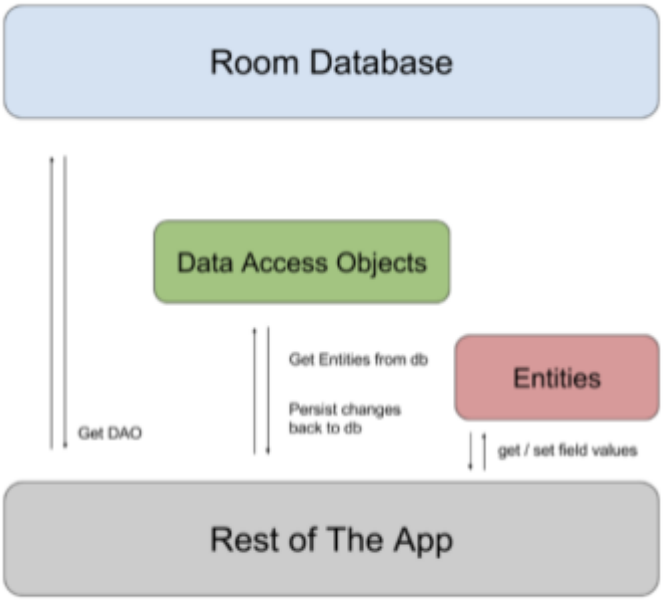
Architecture

Since the application should run on Android phones, there are languages selectable; Java or Kotlin. For this course Java was set in the requirements. The most common architecture for Android mobile applications is the Clean Architecture. This application contains three application layers [R1]:

1. Presentation Layer
2. Business or Domain Layer
3. Data Layer

In the presentation layer where the user interface and the user experience resists are the different screens from above and the Fragment classes in the class diagram. There are additional Fragments for the different categories in the survey. The material.io [L3] library is used for the general design. With this library developers can create modern looking designs.

The supporting classes belong to the Business Layer, among them are for example Fragments for the given survey categories. The application has service and receiver classes working together for setting up the timer for notifications to remind the user that the activity recording runs more than 3 hours or to take the survey in the given time that the user can set up in the settings.
The worker classes in this application process the data for storing it in the local database and the cloud storage. Helper classes are used to recognize the change of the accelerometer and the sliders in the surveys. There is also a class which is loading the deviated database and doing the configuration of the statistic chart. The last one is the ServiceNotification helper and displays the recording activity information in the task bar.



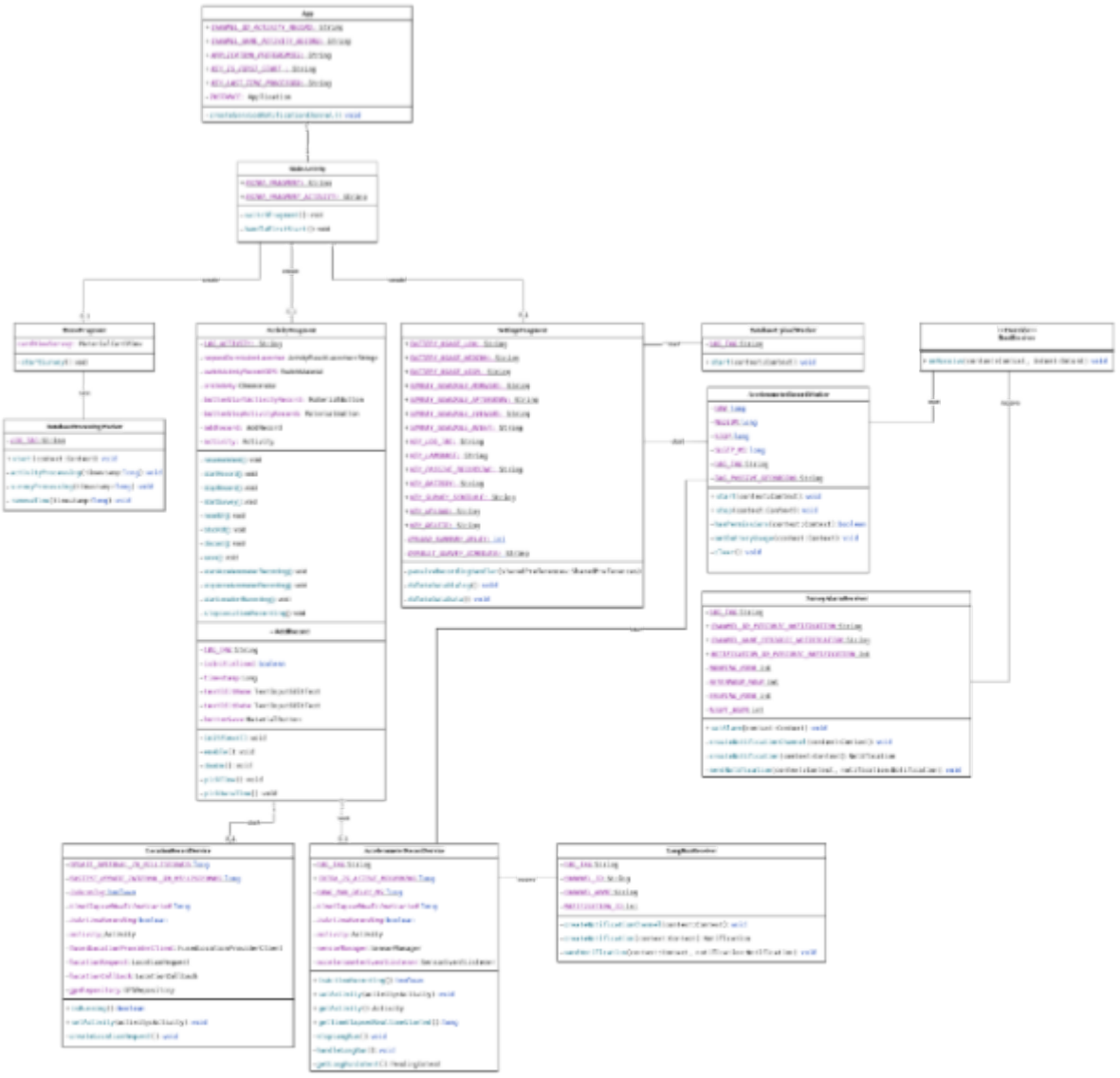
Room library architecture [L1]

All the Room database files (24 files) are in the data layer. Each of the seven databases owns three files. The LocalDatabase and the Research Database manage the Database class files; these in turn manage the persistent data on the phone. The Data Access Objects (DAO) provides methods for data processing. The database tables

are represented by the Entities or in our case Repositories. In addition to this, the Firebase cloud storage keeps the pseudonymized database files of the app users. This means that there is some kind of client server architecture besides the CLEAN - Architecture. It has beneficial aspects like usability, scalability

Description of the class diagram

During the boot process the application loads the key for the notification process of the activity recording, the processing the database, and the application start. An instance of the application will be created in the App.class.
The Main.class contains methods for the fragment (screen)-switching and another method for the first start of the application to load background processes for the navigation of the menu-bar. After the application boot process is finished, the HomeFragment.class will be called. The survey activity will be called within this class to start the activity which is stored in a different class. Therefore the call for processing for the user statistics chart data begins here.
The recording of the activity is done in the ActivityFragment.class which contains methods to start and stop the recording of the accelerometer sensor either with or without the GPS-location sensor. The user interface for the Activity Fragment is blocked during the recording process. After the recording is stopped by the user the Survey Activity will be started so that the user can immediately input his/her mood. If the user forgets to record intense activity he/she can make an entry manually with the activity and the time in the database. Since this is not a measured dataset, this entry does not affect the acceleration or GPS data otherwise it will distort the reality, and we think this is the wrong way of doing research. As discussed in the previous section the application sends a notification using the LongRunReceiver, which is an inheritance of the AccelerationRecordService three hours after the start of the activity recording. The last Fragments contains all the default settings for the acceleration recording on boot or if it is changed then the preference settings. The survey notification broadcast receiver also starts on the boot process.



Class diagram overview

Testing

The testing of the complete app was done gradually and several ways, to be 99.99% sure that the app is fulfilling the requirements and is also working as it should. To start with the testing process, we first of all created for everyone private sub branches from the main branch, in which everyone could carry out his or her task separately.
After testing every individual implementation and being completely certain it doesn't affect the app, we will merge them to the main branch. Since new implementations were constantly integrated into the main branch, we had to always pull the latest version of the app on Github into our private sub branches, merge and test them locally. Furthermore, an "End to End" testing was found suitable for our testing which could be for example an emulator or on native devices. The testing in both cases was carried out differently, since they don't work the same way but carry out the same task.
During the development process debug logging was implemented on critical parts of the program to check the correctness of the methods. For the emulator testing, the internal debugger in Android Studio was constantly used to complete a detailed path of the program each time a bug was located or affecting the program from running. As for the native device testing, we had to test the app on various devices(Samsung, Huawei) with different android versions (from 9 up to 12).
After the deployment in the main branch another test was done and we used the "issue tracker" on Github to identify the bugs(if they were any) and were quickly taken care of.

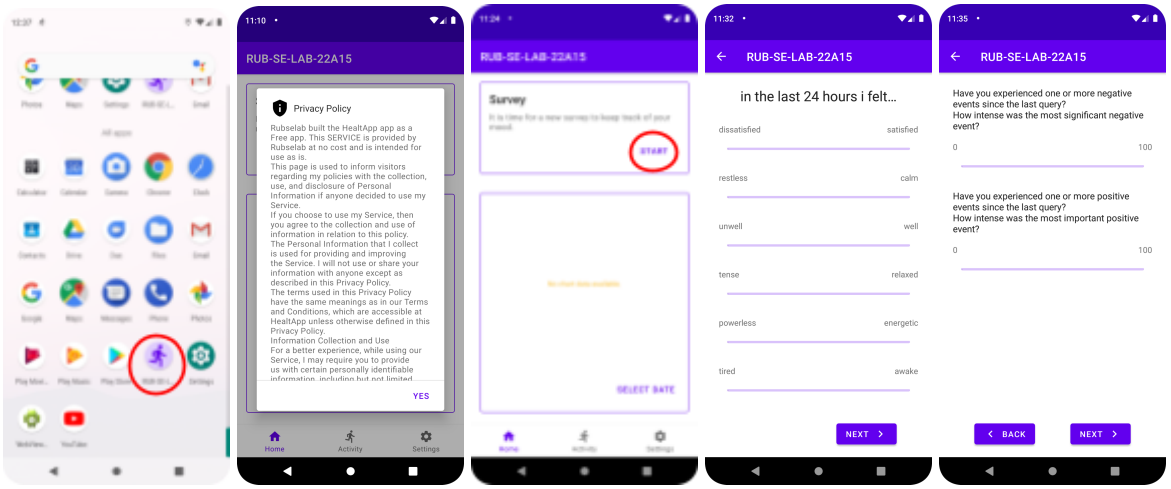
Licenses

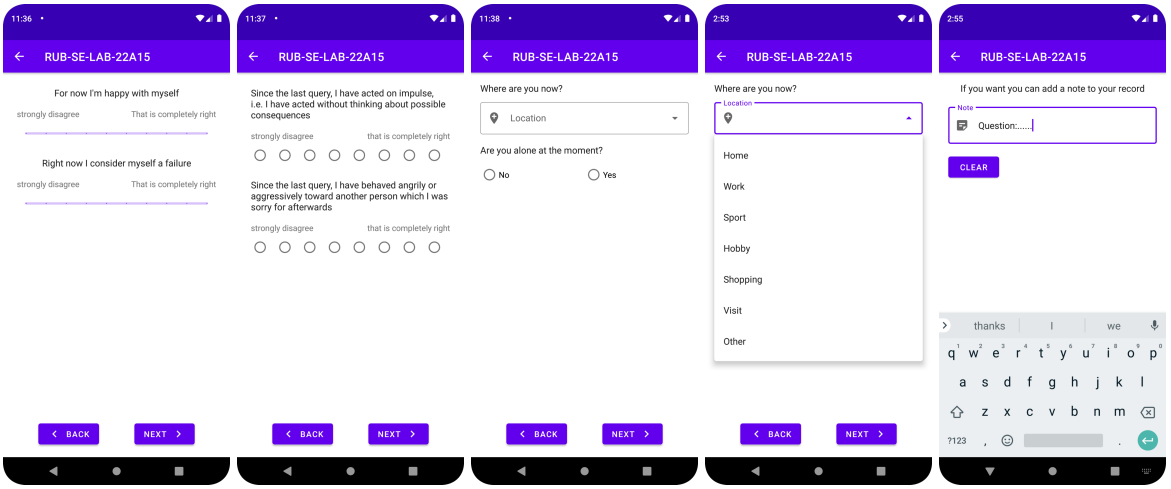
[L1] Room Local Database: https://developer.android.com/training/data-storage/room
[L2] Firebase Cloud Storage: https://firebase.google.com/docs/storage/android/start
[L3] Design: https://material.io
[L4] Chart: https://github.com/PhilJay/MPAndroidChart

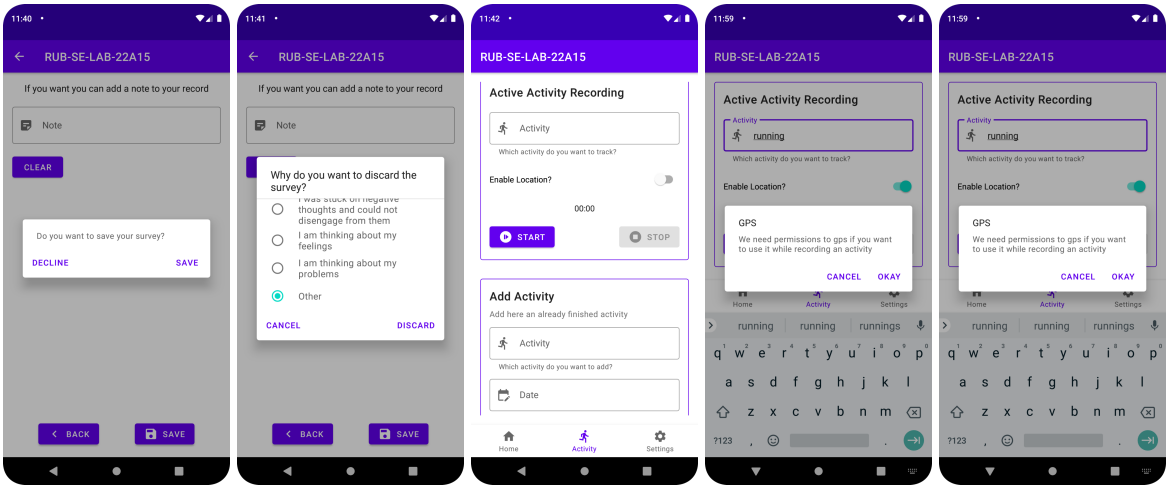
References

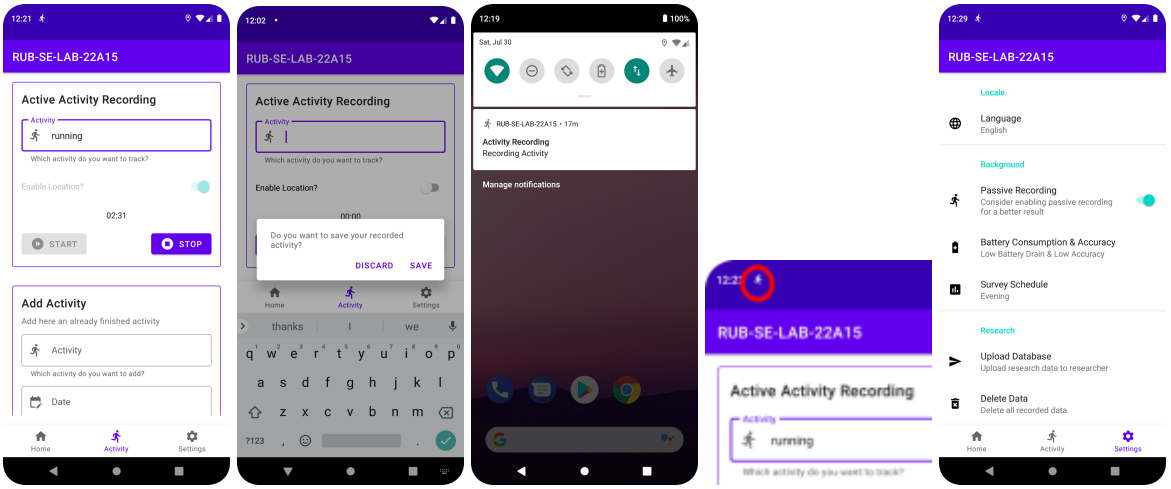
[R1]
https://www.netsolutions.com/insights/mobile-app-architecture-guide/#:~:text=Mobile%20app%20architecture%20refers%20to,as%20well%20as%20industry%20standards.

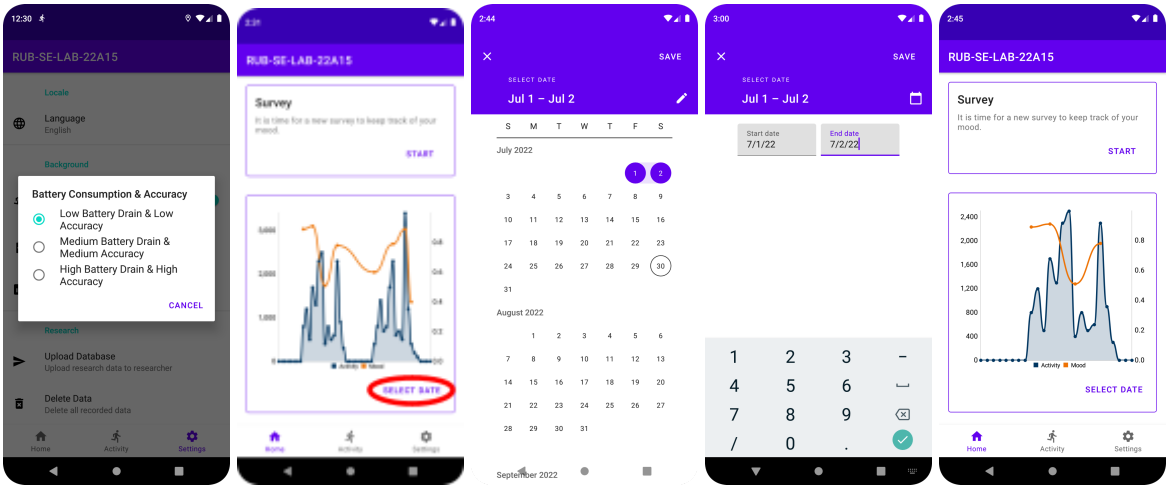
Appendix

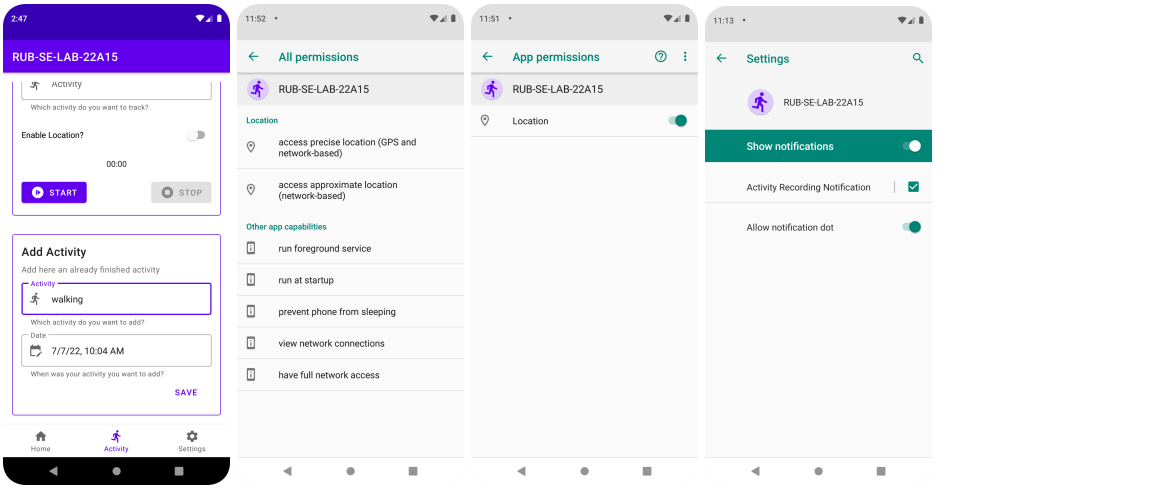


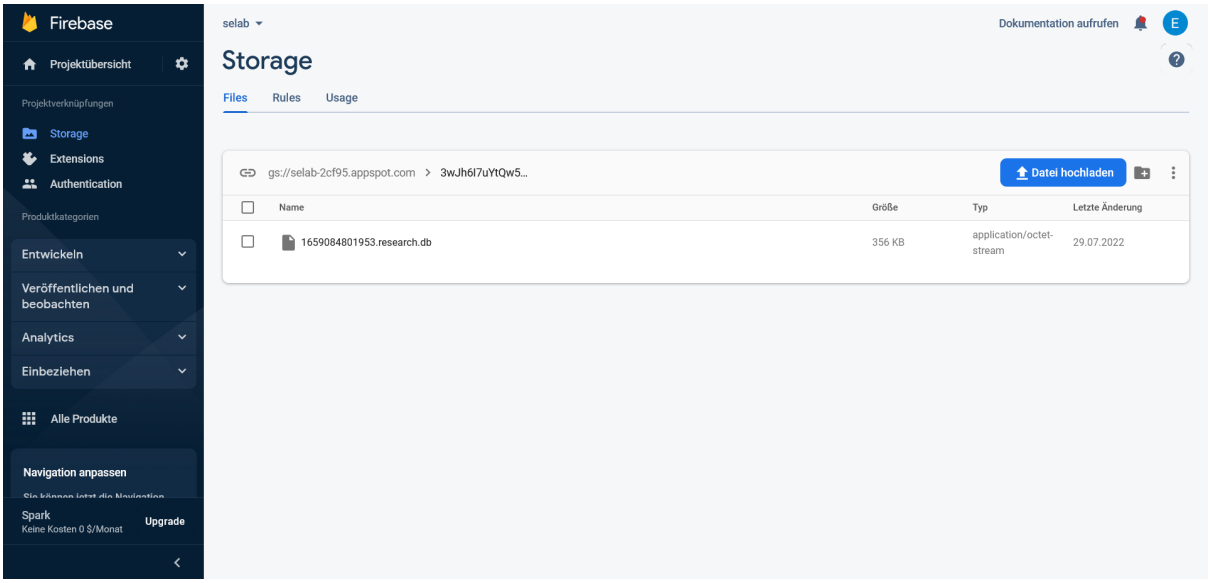


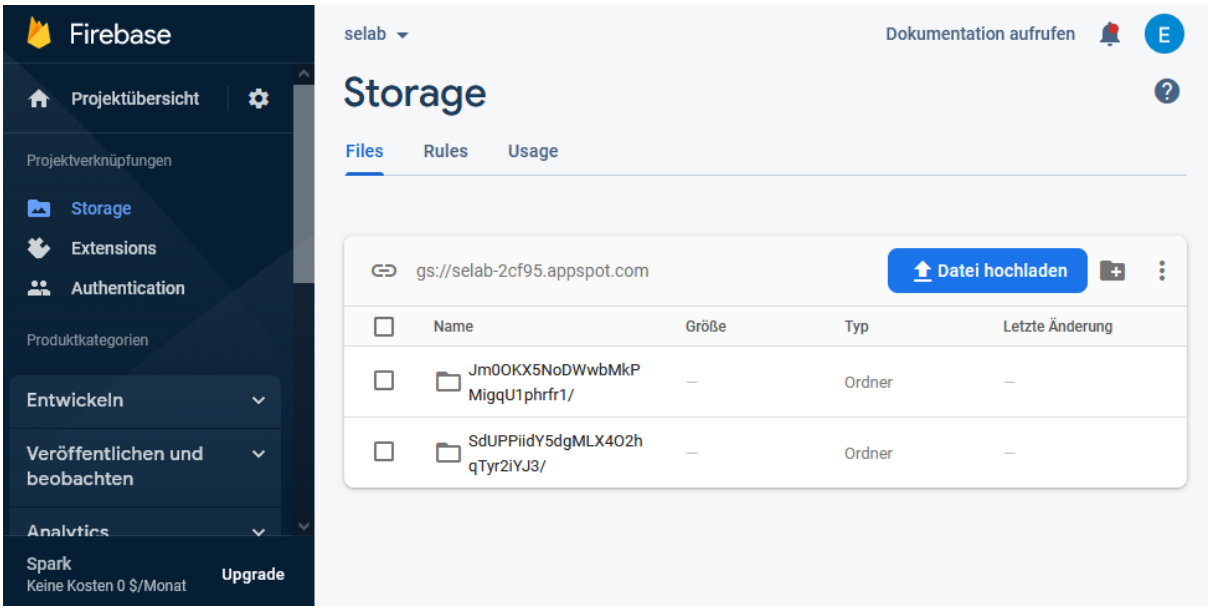


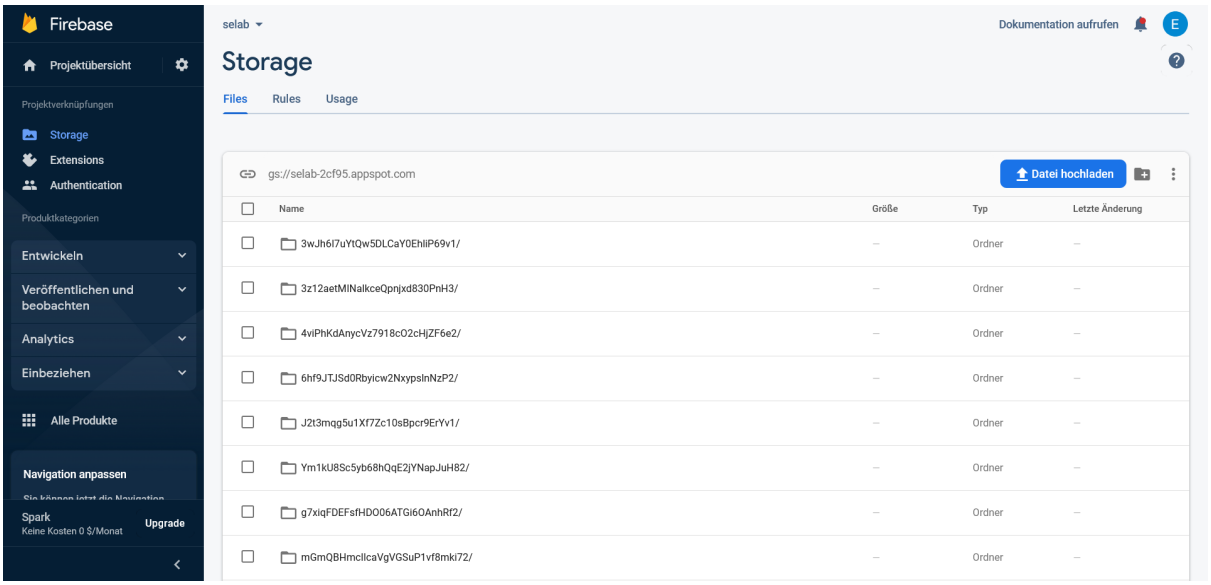

RUB-SE-LAB-22A15
Active Activity Recording
Activity
running
Which activity do you want to track?
Enable Location?
02:31
START
STOP
Add Activity
Add here an already finished activity
Activity
Which activity do you want to add?
Date
Home
Activity
Settings
Do you want to save your recorded activity?
DISCARD
SAVE
Manage notifications
Locale
Language
English
Background
Passive Recording
Consider enabling passive recording for a better result
Battery Consumption & Accuracy
Low Battery Drain & Low Accuracy
Survey Schedule
Evening
Research
Upload Database
Upload research data to researcher
Delete Data
Delete all recorded data

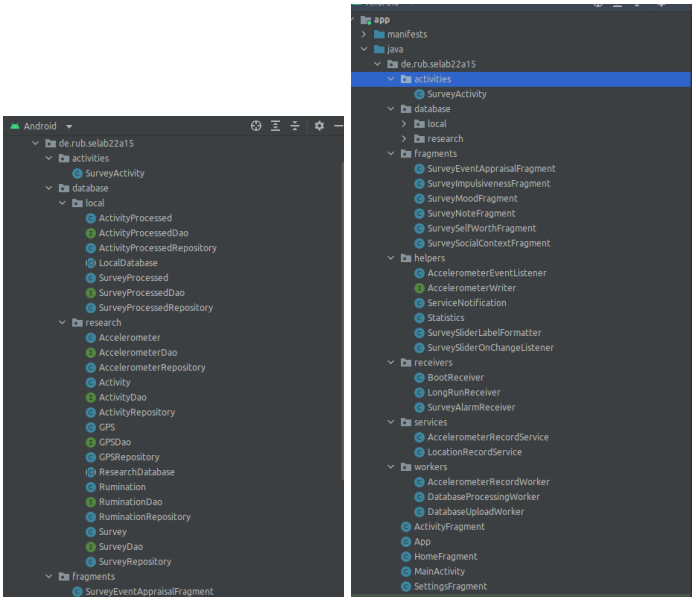

RUB-SE-LAB-22A15
Battery Consumption & Accuracy
Low Battery Drain & Low Accuracy
Medium Battery Drain & Medium Accuracy
High Battery Drain & High Accuracy
CANCEL
Survey
It is time for a new survey to keep track of your mood.
START
SELECT DATE
SAVE
Jul 1 – Jul 2
July 2022
August 2022
September 2022
Start date
7/1/22
End date
7/2/22


RUB-SE-LAB-22A15
Activity
Which activity do you want to track?
Enable Location?
00:00
START
STOP
Add Activity
Add here an already finished activity
walking
Which activity do you want to add?
7/7/22, 10:04 AM
When was your activity you want to add?
SAVE
All permissions
App permissions
Settings
RUB-SE-LAB-22A15
Location
access precise location (GPS and network-based)
access approximate location (network-based)
Other app capabilities
run foreground service
run at startup
prevent phone from sleeping
view network connections
have full network access
Show notifications
Activity Recording Notification
Allow notification dot


Firebase
Projektübersicht
Projektverknüpfungen
Storage
Extensions
Authentication
Produktkategorien
Entwickeln
Veröffentlichen und beobachten
Analytics
Einbeziehen
Alle Produkte
Navigation anpassen
Spark
Keine Kosten 0 $/Monat
Upgrade
selab
Dokumentation aufrufen
Storage
Files
Rules
Usage
gs://selab-2cf95.appspot.com > 3wJh6I7uYtQw5...
Datei hochladen
Name
Größe
Typ
Letzte Änderung
1659084801953.research.db
356 KB
application/octet-stream
29.07.2022


Firebase
Projektübersicht
Projektverknüpfungen
Storage
Extensions
Authentication
Produktkategorien
Entwickeln
Veröffentlichen und beobachten
Analytics
Spark
Keine Kosten 0 $/Monat
Upgrade
selab
Dokumentation aufrufen
Storage
Files
Rules
Usage
gs://selab-2cf95.appspot.com
Datei hochladen
Name
Größe
Typ
Letzte Änderung
Jm0OKX5NoDWwbMkP MigqU1phrfr1/
Ordner
SdUPPiidY5dgMLX4O2h qTyr2iYJ3/
Ordner


Firebase
Projektübersicht
Projektverknüpfungen
Storage
Extensions
Authentication
Produktkategorien
Entwickeln
Veröffentlichen und beobachten
Analytics
Einbeziehen
Alle Produkte
Navigation anpassen
Spark
Keine Kosten 0 $/Monat
Upgrade
selab
Dokumentation aufrufen
Storage
Files
Rules
Usage
gs://selab-2cf95.appspot.com
Datei hochladen
Name
Größe
Typ
Letzte Änderung
3wJh6I7uYtQw5DLCaY0EhliP69v1/
3z12aetMINalkceQpnjxd830PnH3/
4viPhKdAnycVz7918cO2cHjZF6e2/
6hf9JTJSd0Rbyicw2NxypsInNzP2/
J2t3mqg5u1Xf7Zc10sBpcr9ErYv1/
Ym1kU8Sc5yb68hQqE2jYNapJuH82/
g7xlqFDEFsfHDO06ATGi6OAnhRf2/
mGmQBHmcIcaVgVGSuP1vf8mki72/
Ordner


Android
de.rub.selab22a15
activities
SurveyActivity
database
local
ActivityProcessed
ActivityProcessedDao
ActivityProcessedRepository
LocalDatabase
SurveyProcessed
SurveyProcessedDao
SurveyProcessedRepository
research
Accelerometer
AccelerometerDao
AccelerometerRepository
Activity
ActivityDao
ActivityRepository
GPS
GPSDao
GPSRepository
ResearchDatabase
Rumination
RuminationDao
RuminationRepository
Survey
SurveyDao
SurveyRepository
fragments
SurveyEventAppraisalFragment
app
manifests
java
de.rub.selab22a15
activities
SurveyActivity
database
local
research
fragments
SurveyEventAppraisalFragment
SurveyImpulsivenessFragment
SurveyMoodFragment
SurveyNoteFragment
SurveySelfWorthFragment
SurveySocialContextFragment
helpers
AccelerometerEventListener
AccelerometerWriter
ServiceNotification
Statistics
SurveySliderLabelFormatter
SurveySliderOnChangeListener
receivers
BootReceiver
LongRunReceiver
SurveyAlarmReceiver
services
AccelerometerRecordService
LocationRecordService
workers
AccelerometerRecordWorker
DatabaseProcessingWorker
DatabaseUploadWorker
ActivityFragment
App
HomeFragment
MainActivity
SettingsFragment