SLEEP CYCLE USING ANDROID STUDIO (JAVA)

A mini project report submitted by

VISHAL THAMPY BIJU (URK18CS175)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

under the supervision of

Mr. Radhakrishnan, Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed-to-be-under Sec-3 of the UGC Act, 1956)

Karunya Nagar, Coimbatore - 641 114. INDIA

March 2021

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
A CHRISTIAN MINORITY RESIDENTIAL INSTITUTION
AICTE Approved & NAAC Accredited

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project report entitled, "**Sleep Cycle using Android Studio**" is a bonafide record of Mini Project work done during the even semester of the academic year 2020-2021 by

VISHAL THAMPY BIJU (Reg. No: URK18CS175)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Karunya Institute of Technology and Sciences.

Submitted for the Viva Voce held on **29 - March – 2021.**

Project Coordinator **Signature of the Guide**

ACKNOWLEDGEMENT

First and foremost, I praise and thank ALMIGTHY GOD whose blessings have bestowed in me the will power and confidence to carry out my project.

I am grateful to our beloved founders Late**. Dr. D.G.S. Dhinakaran, C.A.I.I.B, Ph.D.** and **Dr. Paul Dhinakaran, M.B.A, Ph.D.**, for their love and always remembering us in their prayers.

I extend my thanks to our Vice Chancellor **Dr. P. Mannar Jawahar, Ph.D.** and our Registrar **Dr. Elijah Blessing, M.E., Ph.D.,** for giving me this opportunity to do the project.

I would like to thank **Dr. Prince Arulraj, M.E., Ph.D.,** Dean, School of Engineering and Technology for his direction and invaluable support to complete the same.

I would like to place my heart-felt thanks and gratitude to **Dr. J. Immanuel John Raja, M.E., Ph.D.,** Head of the Department, Computer Science and Engineering for his encouragement and guidance.

I feel it is a pleasure to be indebted to, **Mr. J. Andrew, M.E, (Ph.D.),** Assistant Professor, Department of Computer Science and Engineering and **Mr. Radha Krishnan, Assistant Professor (Guide)** for their invaluable support, advice and encouragement.

I also thank all the staff members of the Department for extending their helping hands to make this project a successful one.

I would also like to thank all my friends and my parents who have prayed and helped me during the project work.

ABSTRACT

Maintaining good sleep hygiene is a constant challenge in modern lives. Sleep habits are hard to monitor and record, especially when most sleep monitoring programs overlook the necessity of calculating user input. This input is vital in order to change poor sleeping patterns, as it is difficult to identify the source of an individual's problems. Sleep tracking software also struggle with a lack of user transparency and interactivity leading individuals to mistrust the results these applications generate or otherwise not feel like the insights are actionable.

To explore these issues, we designed an interventional chat bot to mediate information collection and interaction between end user and sleep monitoring technology. The Sleep Bot prompts users with simple questions that attempt to elicit insight into larger problems that contribute to poor sleep and help craft successful sleep hygiene behaviors. Text messaging-based interaction eases the process as it is similar to talking with a friend, making for a unique environment in which the user is able to share personal data comfortably.

Smartphones were originally mainly used for making phone calls and playing games, but as they become more powerful and are equipped with a wide variety of sensors new use cases become interesting. One of these use cases is sleep monitoring, which is interesting for many different research areas. The goal of this bachelor thesis is to develop a sleep monitoring framework for the Android platform which can be used easily by third party applications. The framework takes care of detecting sleep related events like snoring and movement as well as monitoring the ambient light during the night. Additionally, a demo application is developed to demonstrate the functionality of the framework and to highlight some best practices regarding Android background services as they are essential for monitoring sleep.

CONTENTS

1 Introduction

As the amount of people owning a powerful mobile device grows, the urge to develop software which helps to handle the daily routine as well as improving it, increases as well. Almost every modern smartphone is equipped with a variety of sensors, ranging from cameras, light sensors and microphones to motion sensors and compasses. Thanks to the hardware and operating system developers it becomes increasingly easy to use those sensors in custom applications, thus helping users to better understand their body and helping them to live a more efficient and happy life.

Sleep is an essential part of our lives; when not being able to sleep well or long enough, we feel tired, are less productive and are likely to gain weight and as a result even develop major diseases like diabetes [16]. Moreover, we need the time at night to come to rest and to give our brains the possibility to process the things we experienced throughout the day. Additionally, problems or irregularities regarding sleep often originate from other diseases.

Therefore, being able to monitor sleep without the need of a conventional sleep study could help in a wide variety of use cases. Conventional sleep studies cost up to $1000-$5000 [20][15][19] whereas using a smartphone application only requires the user to download it once and remember to start and stop it in the evening and morning. As 80% of adults which are regularly online also own a smartphone, most participants already have their monitoring device at their fingertips [13]. Of course a smartphone application can’t provide the detailed insights of an extensive sleep study but having sleep data of a wide variety of patients with very low cost is a huge benefit for sleep researchers.

1.1 Objective

The objective of this app is to monitor your sleep pattern and suggest you tips for your better health.

Sleep Tracker includes two main features: debt estimation and "smart alarms".

- Sleep Tracker can estimate how much sleep debt the user has incurred.
- Modern devices running Android and iOS include built-in microphones and accelerometers. Sleep Tracker "smart alarm" feature is designed to attempt to wake the user during non rem

sleeping. This does not prevent sleep debt or prevent the user from experiencing any of the mental and physical side effects of sleep desperation, but it may help to prevent early-morning grogginess and sleep.

Sleep Tracker also includes a variety of minor features, including sound recording, motion graphing, gentle alarms, sleep trend graphs, bedtime reminders, and others.

- If the app's "record sound" and "track motion" features are enabled, Sleep Tracker will use a device's microphone and accelerometer (if available) to record noise and movement while the user sleeps. Upon arising, the user may view sound and movement graphs, and may tap on certain parts of the sound graph to hear snippets of sound.
- Sleep trend graphs show what time the user went to sleep, what time the user woke up, how many hours the user slept, and other data.
- The app's "smart alarms" can be configured to ring gently. Gentle alarms start out quietly and take five minutes to ramp up to full volume.
- At night, Sleep Cycle can silence a device, disable WIFI and enable airplane mode. These three features, when used together, will prevent a device from receiving calls, text messages, and other notifications.
- If properly configured, the app can provide bedtime reminders every night, to remind the user to go to bed.
- The app and the Sleep Cycle website each include a "Resources" section with advice regarding sleep and insomnia.

1.2 Motivation

Creating a sleep monitoring application for Android might seem easy at first sight but presents some difficulties at closer look. Simply reading audio data from the internal microphone or fetching light intensity is easy but interpreting it in the context of sleep monitoring is non-trivial. This thesis explores the possibilities of using Android device sensors for sleep monitoring and describes the development and usage of a sleep monitoring framework. The aim of the framework is to provide a high level API for accessing sleep related data and taking care of recording and saving the necessary data.

Providing such a framework enables the development of a 1 wide variety of applications without them having to worry about audio and light interpretation, long running background services or CPU usage and power consumption. Therefore, the development of specific applications for specific use cases becomes only a matter of embedding the library and creating a user interface which might provide some additional use case specific functions. A simple implementation of the developed framework is also discussed, so the aforementioned mobile applications may use it as a starting point. The framework is designed for very easy usage. Especially the storage and retrieval of the collected and interpreted data is very easy to adapt to third party applications, so they can use the framework while keeping their existing data storage components and being able to upload or export the data as needed.

1.3 Over View of the Project:

Sleep Cycle is a time calculator app which enables users to manage their sleep cycles in a better way.

The calculations are based on various studies which talk about the sleeping patterns of the human brain. The app offers features like ‘Fixed Wake-up,’ which enables users to specify the times for waking up.

It also suggests the best time to go to sleep. Along with this, Sleep cycle comes with a ‘Sleep Now’ feature which suggests the best time to set alarm clock, so that you wake up fresh.
It also provides detailed analysis of your sleep by tracking factors like sleep cycles, resting heart rate, respiration and more. Its smart alarm feature can wake you up from a slight sleep. Moreover, Sleep Cycle offers daily tips regarding sports, exercise, as well as stress and weight management.

2. Analysis and Design

Analysis” is a broad term, best qualified, as in requirements **analysis** (an investigation of the requirements) or object **analysis** (an investigation of the domain objects). **Design** emphasizes a conceptual solution that fulfills the requirements, rather than its implementation

2.1 Functional Requirements

There are two basic types of sleep: rapid eye movement (REM) sleep and non-REM sleep (which has three different stages). Each is linked to specific brain waves and neuronal activity. You cycle through all stages of non-REM and REM sleep several times during a typical night, with increasingly longer, deeper REM periods occurring toward morning.

Stage 1 non-REM sleep is the changeover from wakefulness to sleep. During this short period (lasting several minutes) of relatively light sleep, your heartbeat, breathing, and eye movements slow, and your muscles relax with occasional twitches. Your brain waves begin to slow from their daytime wakefulness patterns.

Stage 2 non-REM sleep is a period of light sleep before you enter deeper sleep. Your heartbeat and breathing slow, and muscles relax even further. Your body temperature drops and eye movements stop. Brain wave activity slows but is marked by brief bursts of electrical activity. You spend more of your repeated sleep cycles in stage 2 sleep than in other sleep stages.

Stage 3 non-REM sleep is the period of deep sleep that you need to feel refreshed in the morning. It occurs in longer periods during the first half of the night. Your heartbeat and breathing slow to their lowest levels during sleep. Your muscles are relaxed and it may be difficult to awaken you. Brain waves become even slower.

2.2 Non-Functional Requirements

REM sleep first occurs about 90 minutes after falling asleep. Your eyes move rapidly from side to side behind closed eyelids. Mixed frequency brain wave activity becomes closer to that seen in wakefulness. Your breathing becomes faster and irregular, and your heart rate and blood pressure increase to near waking levels. Most of your dreaming occurs during REM sleep, although some can also occur in non-REM sleep. Your arm and leg muscles become temporarily paralyzed, which prevents you from acting out your dreams. As you age, you sleep less of your time in REM sleep. Memory consolidation most likely requires both non-REM and REM sleep.

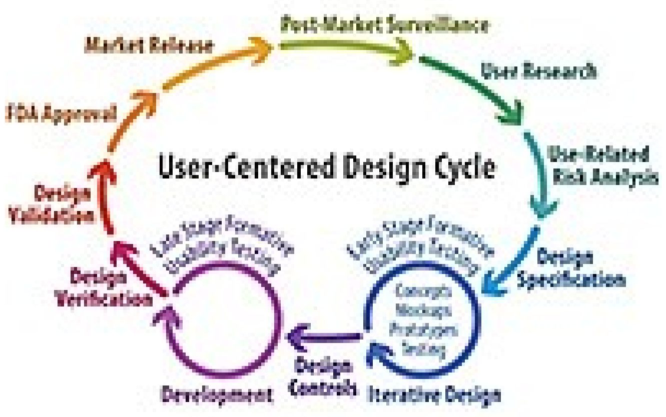
2.3 Architecture

Sleep architecture refers to the basic structural organization of normal sleep. There are two types of sleep, non-rapid eye-movement (NREM) sleep and rapid eye-movement (REM) sleep. NREM sleep is divided into stages 1, 2, 3, and 4, representing a continuum of relative depth. Each has unique characteristics including variations in brain wave patterns, eye movements, and muscle tone. Sleep cycles and stages were uncovered with the use of electroencephalographic (EEG) recordings that trace the electrical patterns of brain activity (Loomis et al., 1937; Dement and Kleitman, 1957a).
In this project the architecture is to tale the input from the user analyse it and the give suggestions.

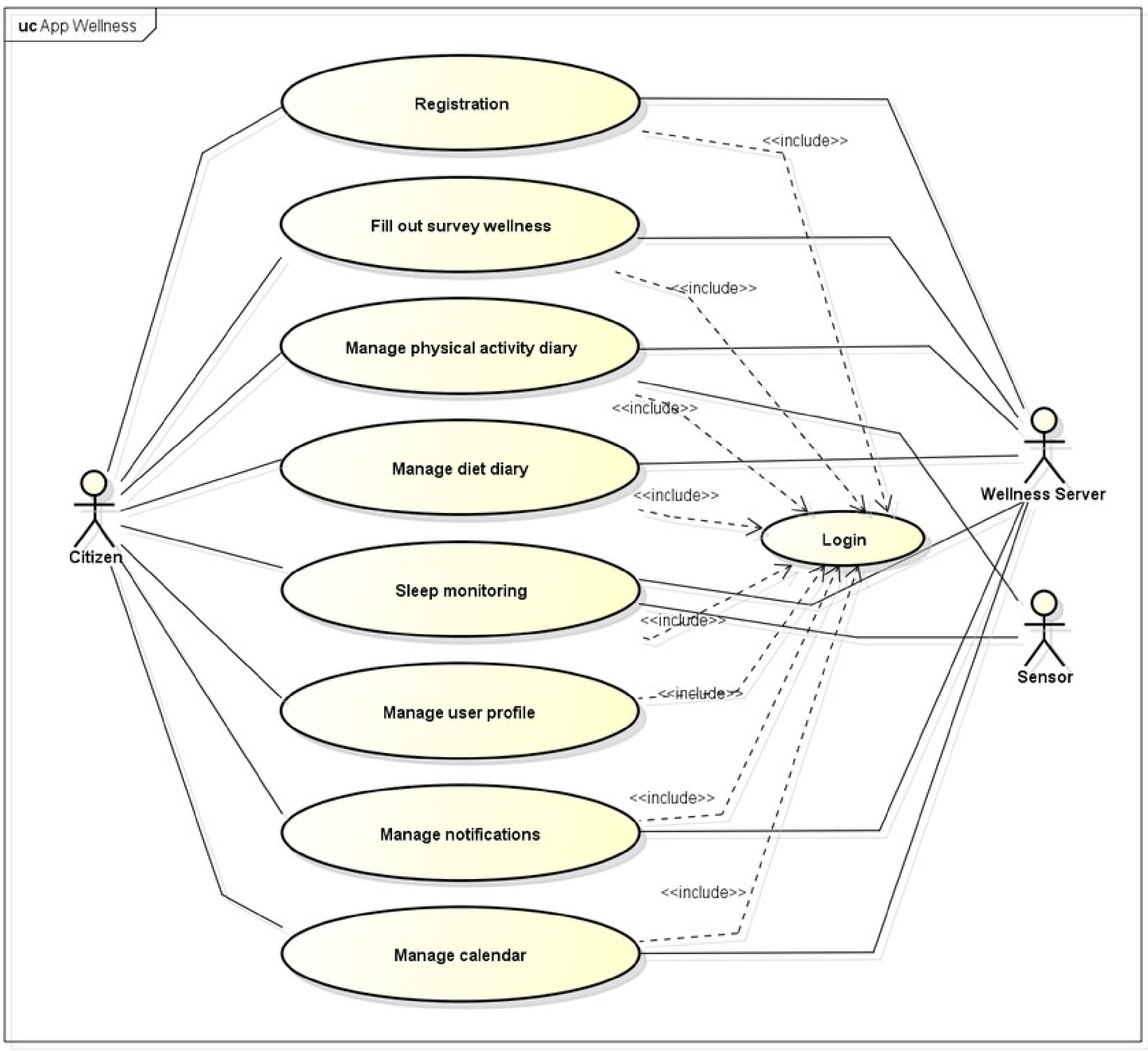
2.4 Use case diagram

“It can shine a light on your overall sleep patterns,” Schwartz says. You’ll gain insight on your habits and how they affect your slumber. You can use that device to your advantage in the following ways.

- Get a general idea of how much you’re sleeping. More than one-third of Americans don’t get enough shut-eye on a regular basis. But many don’t realize it. “A sleep tracker can give you some awareness whether you’re close to the recommended amount of sleep,” Gooneratne says. Although those numbers may not be 100% accurate, you can use them as a rough guide.
- Notice big changes. If your data reveal that you started moving around a lot, mention it to your doctor. That may be a sign of a sleep issue, such as apnea or restless legs syndrome, Simmons says.
- Look for patterns. You can notice trends, such as how much you snooze on weekdays vs. weekends. Or what time is the best bedtime. “I realized that I’m never going to get 7 hours of sleep if I don’t go to bed before 10:30 on workdays,” Baron says. “Of course, I knew that, but seeing it in my data is another story.”



2.5 Sequence Diagram



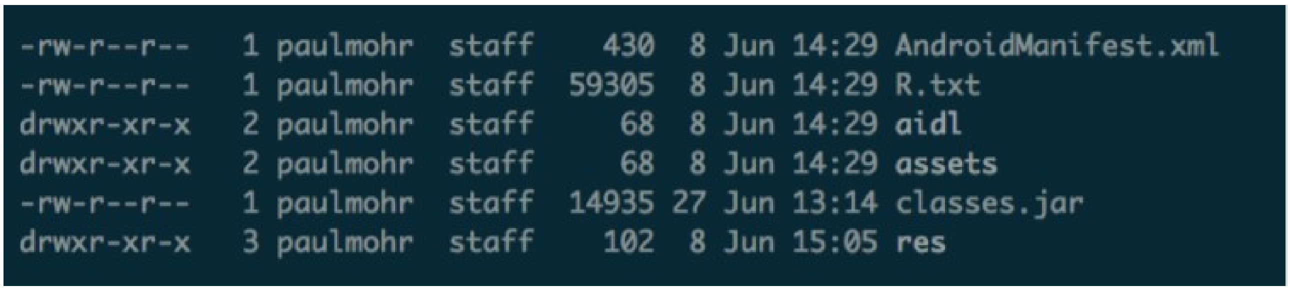
Sequence diagrams can represent a progression of connections between classes or object instances occurrences over the long run. Sequence diagrams are regularly used to show the preparing portrayed being used case situations. Practically speaking, grouping outlines are gotten from use case examination and are utilized in frameworks configuration to infer the connections, connections, and techniques for the items in the framework. Sequence diagrams are utilized to show the general example of the exercises or cooperation's in a utilization case. Each utilization case situation may make one Sequence diagrams, despite the fact that Sequence diagrams are not generally made for minor situations.

3. Implementation

Both, the framework and the demo application have been programmed in Java. It would be possible to use non-native technology, namely web technology, but using Java ensures best performance and sensor compatibility . All code has been written with Android Studio. Android Studio is a very good choice because it has all the tools available to develop a library separated from an Android application while still using them in a single project. It also provides very convenient methods to create many different virtual devices and test the application on them. The application was mainly developed and tested on a real Nexus 6 device but a lot of different virtual Android devices have been used to test the application on the most important Android versions between API level 10 and 21. Figure 4 shows how the framework, the demo application and the sensors work together when the application is recording. All hardware interaction as well as the interpretation of the audio signals is done by the framework and is then saved by the demo application to the device's storage. Due to the later discussed OutputHandler interface, the data could as well be saved directly to an external web service.

3.1 Using an Android framework

While one would assume that using an Android framework is the same as using any .jar file while developing Java applications there are indeed some differences. Android frameworks are not distributed as .jar files but as Android specific .aar files.



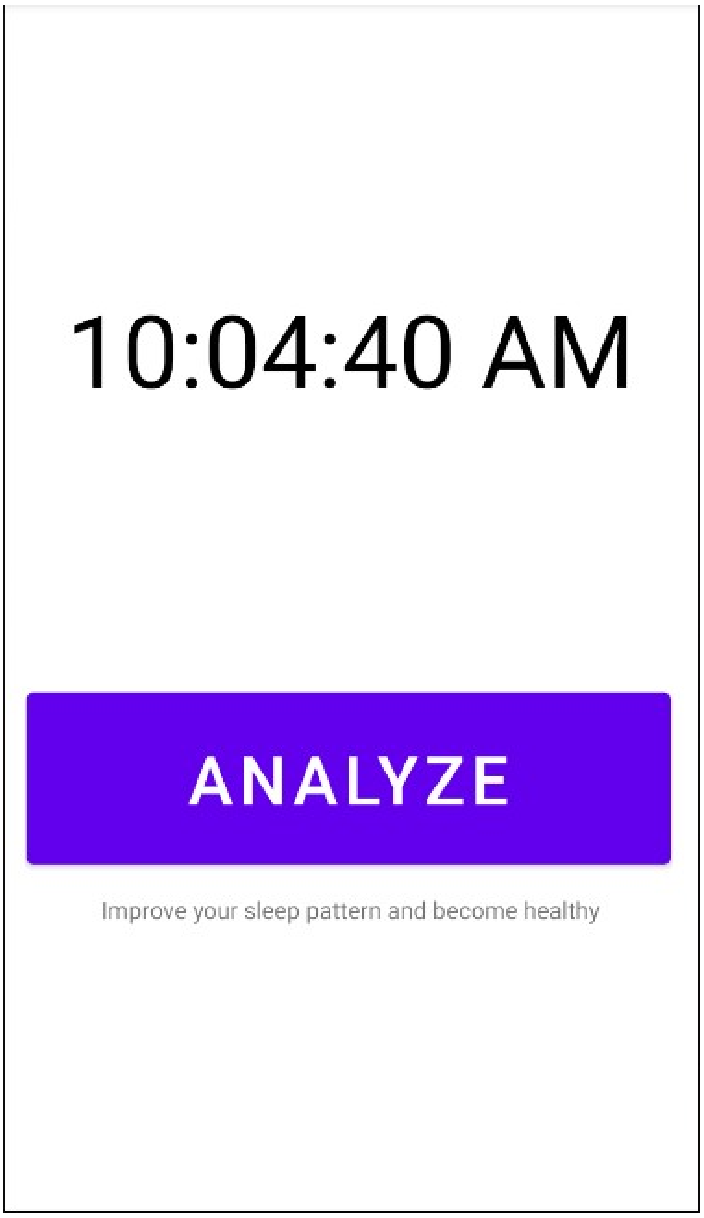
```
-rw-r--r--   1 paulmohr  staff    430  8 Jun 14:29 AndroidManifest.xml
-rw-r--r--   1 paulmohr  staff  59305  8 Jun 14:29 R.txt
drwxr-xr-x   2 paulmohr  staff     68  8 Jun 14:29 aidl
drwxr-xr-x   2 paulmohr  staff     68  8 Jun 14:29 assets
-rw-r--r--   1 paulmohr  staff  14935 27 Jun 13:14 classes.jar
drwxr-xr-x   3 paulmohr  staff    102  8 Jun 15:05 res
```

The differences to a simple .jar file are obvious. The .aar file contains Android specific files and folders like the AndroidManifest.xml file which can be used to define necessary permissions which are needed by the library. These permissions will then be merged into the application's AndroidManifest.xml file. The .aar file can simply be imported into the IDE. For the widely used IDE Android Studio this is fairly easy

```
...

dependencies {

    ...

    compile project(':sleepminder.lib')

}
```

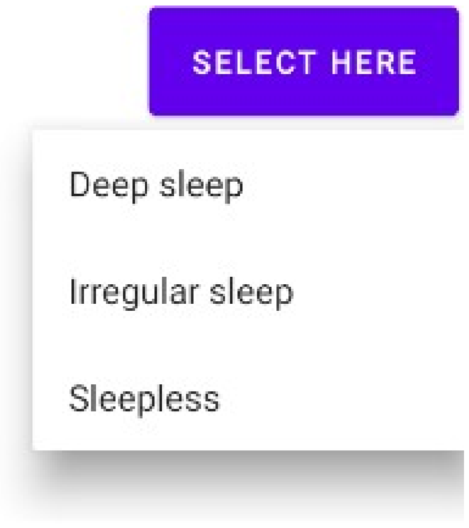
3.2 Demo Application Layout

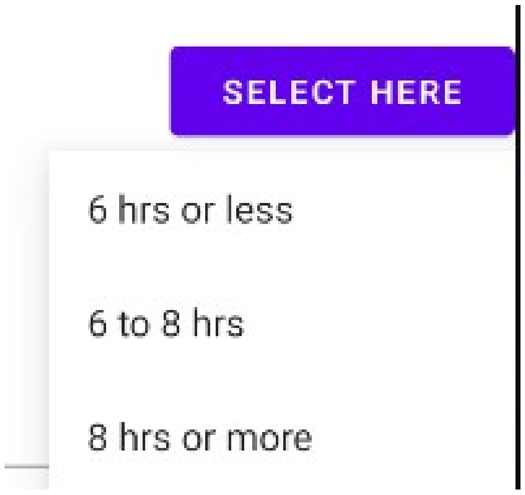
The layout of the demo application is centered around two goals. The application should be very fast to use and it should be usable in dark environments. As it is used mainly in the evening and morning this is a convenient feature for the user. The main screen of the application is presented in figure 16. The primary action is the start / stop button. By tapping on it the user can start the recording. Below the button is a list of past recordings which can be viewed by tapping on them. The used colours can be seen in figure 13. For large areas like the list background and the header the dark shades are used. Therefore the user is not blinded when using the app. When the user starts / stops the recording a popup informs the user about the success of the action. Both, the floating action button and the popup are features of the Android Design Support Library [11]. This library is used to bring the latest layout features to older Android devices as well. Otherwise, the minimum Android version would have to be 5.0 (API level 21) or a separate layout would have to be made for pre Lollipop devices.



3.3 Android Background Service

Even though, it would be possible to start the recording in the main application, there is a much better method which ensures that the recording is not interrupted. The Android Service class is made for exactly this reason. ”A Service is an application component that can perform long running operations in the background and does not provide a user interface” [27]. The Service is started and controlled by an activity and may provide a notification to the user about the current state of the service. The notification which is shown in the sample application can be seen in figure 14. Figure 14: The notification of the demo application in the notification bar In order to create a Service some configuration steps are necessary, so the service behaves as required.

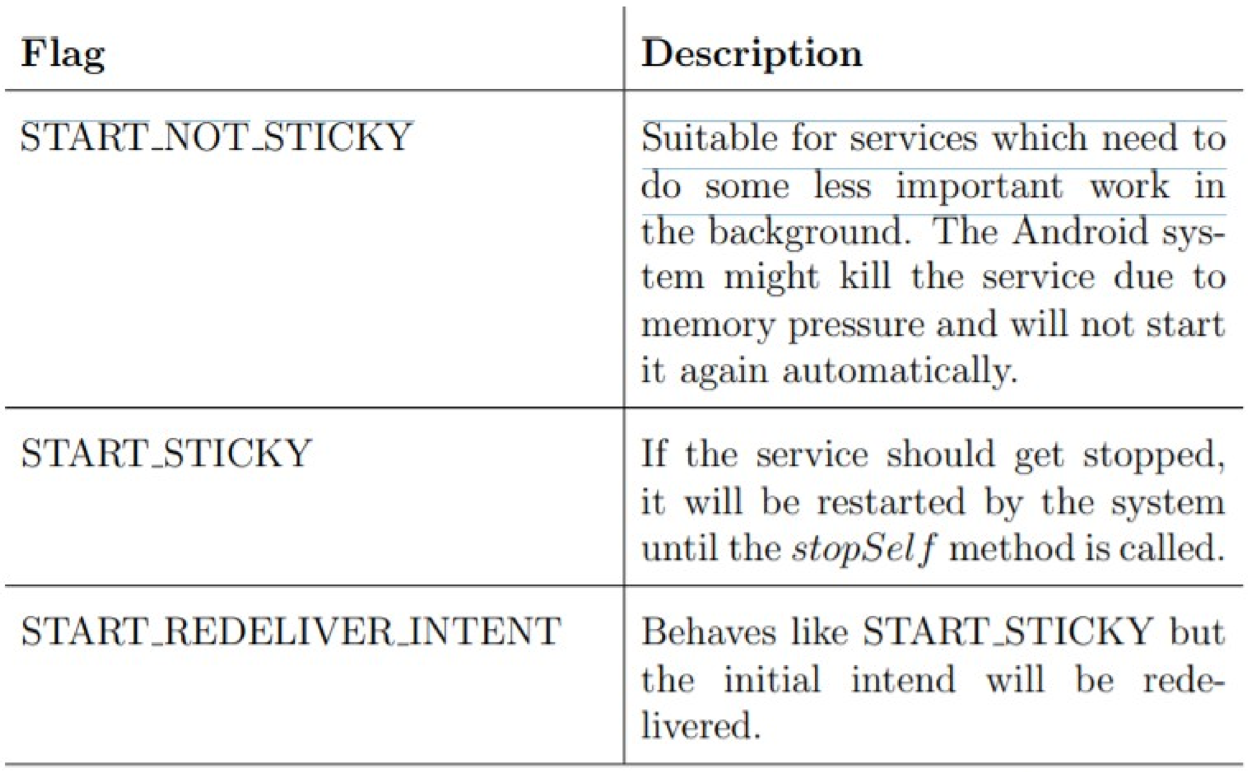




3.4 Declare the Service

Like an activity, the service must be declared in the AndroidManifest.xml file. No further configuration is needed at this place.

Start the Service The startService method must be called on an Android context to start a service. This results in the Android system calling the onStartCommand method of the implemented Service class. The return value of this method is very important as it tells the Android system how to handle the service. Three main return values are available, they are listed in table 6.



Flag	Description
START_NOT_STICKY	Suitable for services which need to do some less important work in the background. The Android system might kill the service due to memory pressure and will not start it again automatically.
START_STICKY	If the service should get stopped, it will be restarted by the system until the *stopSelf* method is called.
START_REDELIVER_INTENT	Behaves like START_STICKY but the initial intend will be redelivered.

Table 6: Some of the return values for the onStartCommand method [26].

In the demo application the START STICKY flag is returned, because keeping the service alive is the number one priority. Redelivering the intent is not necessary in this case as the intend doesn't hold any relevant information, apart from the service itself.

3.5 Data storage

As discussed in section 4.1.3 a custom StorageHandler must be implemented. The demo application stores the recordings in the external storage of the Android device. An alternate solution would be the internal storage which is guaranteed to be available at all times. However, using the internal storage also has some drawbacks. Many users prefer to store as much data as possible on the external storage because the internal storage is usually quite limited. In contrast, the external storage is usually a lot larger and as the recordings do use a good amount of space when the application is monitoring sleep every day, the probability of running out of space is a lot smaller when using the external storage. Furthermore, sharing files from the internal storage is relatively difficult compared with sharing files from the external storage. Another difference between external and internal storage is privacy. It is a lot more difficult to access data stored on the internal storage — even with physical access to the device. Therefore, the internal storage is usually used to store very sensitive information. As the recordings do not save any specific data like sound samples they are not considered very sensitive. Hence, the demo application uses the external storage to persist the recordings.

3.6 Data visualization & interpretation

In figure 17 and 18 the detail view of a recording can be seen. The course of the night is displayed at the top. The time span is shown above the graph, the bars indicate movement. The night is separated into 30 minute intervals. For each interval all movement events of the 5 second intervals are accumulated. If more than one movement event occurred, the interval is considered a light sleep phase. The height of the bars indicate the intensity of the movement. This is quite a large abstraction but worked in practice. Further optimisations could be made to detect the REM / non-REM phases more accurately. Figure 17 shows a pretty normal night. 5-6 sleep phases occurred; usually a single cycle lasts about 100 minutes [22]. Therefore, about 5 phases are normal for a sleep duration of 8 hours. Both screenshots are made on a Nexus 6 device which suffers from the light sensor problems described in section 4.1.1. The light sensor did report a change in the light intensity at about 7:30 because another application requested a SCREEN DIM WAKE LOCK. On a device which keeps the 28 light sensor active, even when the screen is turned off, the ascent of the light intensity would be more slowly

3.7 Sleep stages

This pie chart shows the amount of time spent in light sleep versus the amount of time spent in deep sleep. It was found that for a sleep duration of about 8 hours about 90 minutes in the deep sleep phase (also called slow-wave sleep) are normal [12]. The graph shows a deep sleep percentage of 33% which indicates good sleep.

3.8 Light quality

The light intensities are divided into three groups: Night, dawn and day. Lux values up to 20 are considered as night, values up to 100 as dawn and everything above as daylight. Starting at about 100 lux, light is considered to interfere with the sleep cycle and therefore to affect the sleep in a negative way [29]. Due to the Nexus 6 suffering from the light sensor bug, the light quality pie chart shows 0% dawn light intensity. A perfect night should consist mainly of light below 20 lux and only gets brighter while waking up.

3.9 Sleep quality

The sleep quality diagram shows the distribution of audio events. Repeated snoring would be visible here.

3. 10 Overall quality indicator

Based on the feedback of some test users a simple sleep rating system has been developed. A smiley face indicates the estimated overall sleep quality. There are three different faces: ”Good”, ”Not too bad” and ”Bad”. Three different indicators are calculated to estimate the quality. Each indicator gets a 1, 0 or -1 depending on how good the sleep is. 1 represents good sleep quality, whereas -1 indicates bad quality. The average is then calculated and the rounded value determines the estimated overall quality. The first indicator is light quality. If at least one hour of sleep was during daytime light the quality is rated ”bad”. If daytime light combined with dawnlight is at least 1.5 hours long, the quality is rated as ”not too bad”. Everything else is rated ”good”. The second indicator is the amount of sleep cycles. As discussed earlier about 5 cycles are normal. More than 10 cycles or less than 4 are considered ”not too bad”, everything else is rated ”good”. The third indicator is sleep duration. Sleep duration is difficult to rate 29 based on a single night. Short sleepers may be perfectly fine with only 5.5 hours of sleep while long sleepers may need 8.5 hours [8]. As the reasons for the amount of sleep an individual needs, are not clear yet, the optimal duration would have to be determined by a questionnaire where the user would have to state the needed duration to be well-rested as well as the family background — families have shown to have a similar need for sleep duration [10]. Such a questionnaire could be implemented in the application as well, but would exceed the scope of this thesis

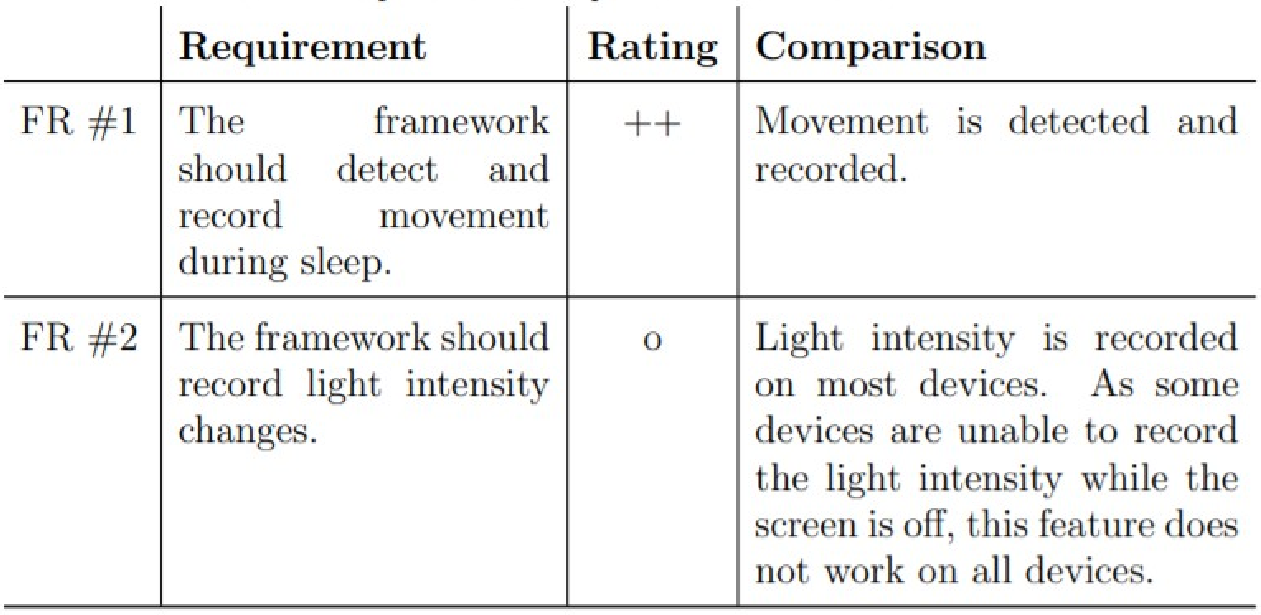
3.11 Power consumption

The user should be able to monitor their sleep without having to connect the smartphone to the power supply. Therefore, one of the secondary goals of the framework and the demo application is to consume as little power as possible. Personal experience showed, that other sleep monitoring applications for Android use about 50-80% of the device’s battery. A small study with six devices has been made to test the power consumption of the demo application. Each device monitored about 8 hours of sleep and was charged very close to 100% before starting the recording. Figure 19 presents the results of the study. Power consumption is between 9% and 16% for all devices. This allows the user to record their sleep without having to worry about power supply.

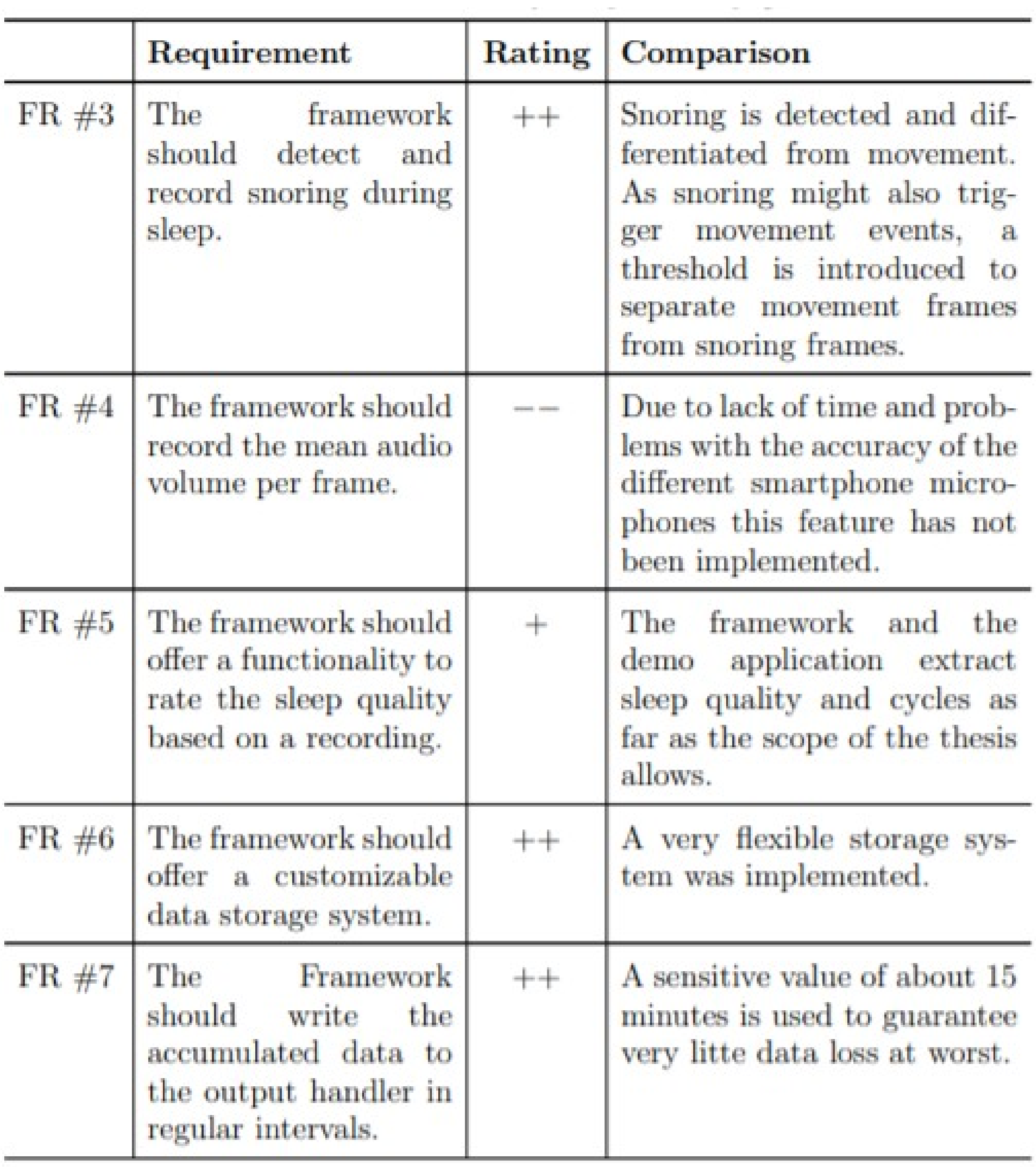
3.12 Framework

Regarding the functional requirements for the framework, not being able to record light intensities on some phones was the main issue. This issue could be resolved partially but still exist on some devices. Apart from that, no major problems remained.

Table 7: Functional requirements comparison for the framework



	Requirement	**Rating**	**Comparison**
FR #1	The framework should detect and record movement during sleep.	++	Movement is detected and recorded.
FR #2	The framework should record light intensity changes.	o	Light intensity is recorded on most devices. As some devices are unable to record the light intensity while the screen is off, this feature does not work on all devices.



	Requirement	Rating	Comparison
FR #3	The framework should detect and record snoring during sleep.	++	Snoring is detected and differentiated from movement. As snoring might also trigger movement events, a threshold is introduced to separate movement frames from snoring frames.
FR #4	The framework should record the mean audio volume per frame.	−−	Due to lack of time and problems with the accuracy of the different smartphone microphones this feature has not been implemented.
FR #5	The framework should offer a functionality to rate the sleep quality based on a recording.	+	The framework and the demo application extract sleep quality and cycles as far as the scope of the thesis allows.
FR #6	The framework should offer a customizable data storage system.	++	A very flexible storage system was implemented.
FR #7	The Framework should write the accumulated data to the output handler in regular intervals.	++	A sensitive value of about 15 minutes is used to guarantee very litte data loss at worst.

4 Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding failures, and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environments
- achieves the general result its stakeholders desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding failures due to software faults. The job of testing is an iterative process as when one fault is fixed, it can illuminate other failures due to deeper faults, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an agile approach, requirements, programming, and testing are often done concurrently.

4.1 Different Types of Software Testing

This article explains only some of the most common types of software testing.

4.1.1. Unit Testing

Testing each component or module of your software project is known as unit testing. To perform this kind of testing, knowledge of programming is necessary. So only programmers do this kind of tests, not testers.

You have to do a great deal of unit testing as you should test each and every unit of code in your project.

4.1.2. Integration testing

After integrating the modules, you need to see if the combined modules work together or not. This type of testing is known as integration testing. You need to perform fewer integration tests than unit tests.

Some good tools for unit and integration testing are Jasmine, Mocha, etc.

4.1.3. End-to-end Testing

End-to-end testing is the functional testing of the entire software system. When you test the complete software system, such testing is called end-to-end testing. You need to perform fewer end-to-end tests than integration tests.

Cucumber, Protractor, Jasmine, Karma, etc. are some great end-to-end testing tools.

4.1.4. User Interface Testing

User interface testing involves the testing of the application’s user interface. The aim of UI tests is to check whether the user interfaces have been developed according to what is described in the requirements specifications document.

By running UI tests, you can make the application’s user interfaces more user-friendly and appealing to the eyes.

Some great automated user interface testing tools are Monkey test for Android, Saucelabs, and Protractor.

4.1.5. Accessibility testing

Testing whether your software is accessible to disabled people or not is termed as accessible testing. For this type of tests, you need to check if disabled people such as those who are color blind, blind, and deaf can use your application.

The right choice of color and contrast need to be made to make your software accessible to color-blind people.

4.1.6. Alpha testing

Alpha testing is a kind of testing to look for all the errors and issues in the entire software. This kind of test is done at the last phase of app development and is performed at the place of the developers, before launching the product or before delivering it to the client to ensure that the user/client gets an error-free software application.

Alpha testing is run before the beta testing, which means that after performing alpha testing, you need to run beta testing.

Alpha testing is not performed in the real environment. Rather, this kind of tests is done by creating a virtual environment that resembles a real environment.

4.1.7. Beta testing

As said earlier, beta testing takes place after alpha testing. Beta testing is done before the launch of the product. It is carried out in a real user environment by a limited number of actual customers or users, in order to be certain that the software is completely error-free and it functions smoothly. After collecting feedback and constructive criticism from those users, some changes are made to make the software better.

So when the software is under beta testing, it is called beta version of the software. After this testing is complete, the software is released to the public.

4.1.8. Ad-hoc testing

As the name suggests, ad-hoc testing is a kind of testing that is performed in an ad-hoc manner, without using any test cases, plans, documentation, or systems. Unlike all other types of testing, this kind of testing is not carried out in a systematic manner.

Although finding errors can be difficult without using test cases, there are technical issues that are easily detected through an ad-hoc test, but are hard to find through other testing approaches that use test cases.

This informal type of software testing can be executed by any person involved with the project.

4.1.9. Compatibility testing

Compatibility testing involves compatibility checking of the software with different operating systems, web browsers, network environments, hardware, and so on. It checks whether the developed software application is working fine with different configurations.

To give you a few examples, if the software is a Windows app, it should be checked whether it is compatible with different versions of the Windows operating system. If it's a web application, it is tested whether the app is easily accessible from different versions of the widely-used web browsers. And if it's an Android app, it should be checked whether it is working well with all the commonly used versions of the Android operating system.

4.1.10. Backward compatibility testing

Backward compatibility testing is carried out to test if a brand new or an updated version of an application is compatible with the previous versions of the environments (such as operating systems and web browsers) on which the software runs. Sometimes, some application is updated specifically to match the standard and style of a newer, more modern environment. In that case, support for backward compatibility is necessary.

Backward compatibility testing ensures that all those who are using the older versions of a particular environment can use your software.

4.1.11. Browser compatibility testing

As the name says, browser compatibility testing checks a web application for browser compatibility. More specifically, it is tested whether the web app can easily be accessed from all versions of the major web browsers.

It is a specific form of compatibility testing, while compatibility testing checks for general compatibility.

Some popular tools to check browser compatibility include CrossBrowserTesting.com, Lambasts, Browser shots, Expedites, Turbo Browser Sandbox, Ranmore Studio, Browser, etc.

4.1.12. Performance testing

Performance tests are run to check if the software's performance is good or not. There are performance testing tools that analyze your app's performance and show you the performance issues. By fixing those issues, you'll be able to increase the performance of your software application.

Some great performance testing tools, also known as load testing tools, for web applications are WebLOAD, LoadView, NeoLoad, LoadNinja, Appvance, LoadRunner, Apache JMeter, Loadstar, Load Impact, Testing Anywhere, SmartMeter.io, Trecentist, Rational Performance Tester, Load Complete, etc.

4.1.13. Load testing

Load testing is one kind of performance testing that tests how much load a system can take before the software performance begins to degrade. By running load tests, we can know the capacity of taking load of a system.

You can run load tests using tools like LoadRunner, Web Load, JMeter, etc.

4.1.14. Recovery testing

Recovery testing involves the checking of whether the application can recover from crashes and how well it recovers. In this kind of tests, testers observe how well the software can come back to the normal flow of execution. Crashes can happen anytime. Even if your software is of exceptional quality, crashes may happen. You don't know when they may take place and annoy the users.

So you have to implement mechanisms that will recover the software application quickly and that will make the application run smoothly again.

4.1.15. Regression testing

If you need to make changes in any component, module, or function, you have to see if the whole system functions properly after those modifications. Testing of the whole system after such modifications is known as regression testing.

4.1.16. Agile testing

Carried out by the QA team, Agile testing is a type of testing that is conducted according to the rules of agile methodology. This kind of testing is done from the actual customers' viewpoint are JIRA, Practices, Junonian, VersionOne, TestRail, SoapUI, etc.

4.1.17. API testing

Just like unit testing, API testing is also a code-level testing type. The basic difference between unit testing and API testing is that unit testing is performed by the development team whereas API testing is handled by the QA team.

4.1.18. Black box testing

Performed by the QA team of a company, black box testing is a testing technique that involves the checking of the application’s functionality without having any technical knowledge of the application, like the knowledge of the code’s logic, how the code works, knowledge of the internal structure, etc.

4.1.19. White box testing

Performed by the development team, white box testing is a testing method that requires a good understanding of the application’s code. It requires great knowledge of the app’s internal logic.

4.1.20. Security testing

Security tests are performed to ensure the security of your application, in order that security breaches can be prevented. Security experts run this kind of tests to see how much your software is secure from attacks and to find security issues so that the app’s security can be strengthened.

4.1.21. Usability testing

Testing the user-friendliness of an app is known as usability testing. It involves the checking of how much usable or user-friendly the app is. It is tested whether any user can easily use your software without getting stuck.

One of the best ways to test the usability of your software is to invite a few people to use your software. See if they can do certain things in your app without taking any help from you.

4.1.22. Scalability testing

Scalability testing verifies whether the software is scalable or not. In other words, it checks if your app performs well when the number of users, amount of data, or the number of transactions increases significantly. A software application that is not scalable may cause great business loss.

4.1.23. Reliability testing

Reliability testing is a type of software testing that verifies if the software is reliable or not. In other words, it checks whether the software runs error-free and that one can rely on it.

For example, if a user's important information stored in the database of the software gets suddenly deleted after a few months because of some error in the code, we can say that the software is not reliable.

4.1.24. Acceptance testing

The client who will purchase your software will perform acceptance testing (also known as User Acceptance Testing) to see if the software can be accepted or not by checking whether your software meets all the client's requirements and preferences. If your software doesn't meet all the requirements or if your client doesn't like something in the app, they may request you to make changes before accepting the project.

5. Closing Statement

The demo application and the framework are a starting point for Android applications which want to collect audio events during human sleep. Although, there are existing Android applications which monitor sleep, none of them are open source or even offer an encapsulated framework. The initial use case for the framework was to develop an application to help diagnosing the correct tinnitus variant. In the final tinnitus app this framework should be used next to other modules. Therefore, the focus to develop an easy to integrate framework was important. Of course, the framework can also be used in other projects. Next to the application and the framework, a complete documentation has been created to enable other developers to use and modify the framework as needed. The documentation is created as Javadoc in the codebase. Additionally, the most important parts are described and explained in detail in this thesis.

6. Acknowledgement

Firstly, I would like to thank my advisor Mr. Radhakrishnan for his support and for giving me the possibility to challenge myself. I would also like to thank my friends for listening to my struggles with motivating me to keep going. Likewise, a lot of thanks to my family for supporting me during the course of my bachelor. Additional thanks go to the Android team for developing such a great mobile platform and especially to the Android Studio team which built a fantastic IDE for Android development.

7. References

https://stackoverflow.com/

https://github.com/federicocotogno/sleepBot

https://www.youtube.com/watch?v=ARezg1D9Zd0