Abstract

Being fatigued is a common state of everyone's daily life. It is a natural sign of your body that signals you to have a break and rest. However, as soon as this state of fatigue lasts longer and does not go back through short breaks, you could consider yourself being cognitively fatigued. This type of fatigue persists after long-term cognitive and mental stress. The purpose of this report is to look at a methodology to use a cross platform application to determine cognitive fatigue.

Introduction

Cognitive fatigue arises from prolonged periods of time in which mentally demanding tasks are performed that induce a state of subjective exhaustion, feelings of "tiredness and lack of energy", and decline in performance [1]. It can be said that mental fatigue is felt much more often after mentally strenuous tasks than after physical exertion. Examples that can lead to cognitive fatigue are, for example, strenuous work days without appropriate compensation or processing the tax return [2]. Especially with prolonged mental stress, we often feel drained and have the feeling that we can no longer continue. A classic example of this is a long drive. This strains our brain so much that after a certain time we are no longer able to drive and need a break, although the body itself did not have to work hard [2]. Symptoms of cognitive fatigue can mainly be described as follows: poor concentration, forgetfulness, slight irritation, simple tasks seem difficult to solve, feeling overwhelmed, physical stress symptoms such as headaches, tension and an increased heart rate, feeling of a mental block, you feel foggy and sleep is harder [3].

Since there is currently no standardized test for recording cognitive fatigue available and biological markers cannot be used to identify these [4], the task was to write an application that should determine cognitive fatigue as part of a study on the basis of daily self-tests.

The application should therefore be able to determine the general condition of a subject with the help of a questionnaire that should be filled out regularly and replace earlier paper questionnaires and be able to save it centrally in a database. In addition, a self-test should be available, which in the best case should be completed several times a day. This should contain two tasks that check the respondent's reaction time and memory ability. In addition, with each run of the self-test, the current emotional state and the current activity of the test person are recorded so that conclusions can be drawn from them later. This information is recorded in a database that is to be exchanged regularly with a server database in order to be able to export all user data centrally for a study.

Background

The project was a study project, which was initially relating to an existing project based on a work from the year 2016 [4]. That means the project is about the implementation of a reusable and expandable platform in order to be able to carry out studies as part of self-tests on cognitive fatigue. To do this, the previous application and all of its properties first had to be examined. From this, some tasks and their expansion options are already known, to which greater attention can be paid during implementation.

Detection of cognitive fatigue

In fact, there are not many ways to recognize cognitive fatigue because markers cannot be evaluated directly [4]. However, two tasks have emerged that can recognize cognitive fatigue based on the speed of reaction and the memory of test subjects. On the basis of previous information from the thesis, two further possibilities are initially not considered. The use of a questionnaire also plays an important role. So far, these have only been evaluated on paper and not electronically.

Psychomotor vigiliance task

A *Psychomotor vigiliance task* (PVT) test is a relatively simple alternative for detecting cognitive fatigue [7]. In PVT tests, the reaction time and attention span are measured, i.e. it is recorded how quickly the person reacts to an event and whether a reaction occurs at all. As a rule, such tests can take about 10 minutes [8].

Spatial Span Task

With the help of the *Spatial span task*, the subject's ability to remember should be tested. For this there is a certain sequence of boxes that light up and the test person should remember. The subject goes through different levels, which become more and more difficult [9].

Questionnaire

The questionnaire is the original variant to illustrate cognitive fatigue. But that was mostly done on paper and then had to be evaluated. These questions should now be included in the application for self-evaluation [4].

Methodology

This section deals with the development process and how the project topic was approached, as well as the decisions made in relation to the programming languages and environments.

Furthermore, we deal with the architecture of the application, that is divided into stand-alone client area and a server architecture. The data exchange and export plays an important role here, which is also discussed in this chapter and later in the implementation part of the report.

Project management

Overall, the project took place within five weeks and five participants who worked full time on the project. The decisions in the area of project management are described in more detail below.

Agile project development with scrum

The project management method SCRUM in a simplified version was chosen for the project. This had the advantage that, on the one hand, programming was agile under certain rules, on the other hand some times and some overhead were reduced.

The Daily Scrum Meeting was an important part of the daily routine, which was occasionally expanded to talk about the current status and follow up on the relevant issues. Due to the limited project time of five weeks, it was decided against the use of sprints and the additional overhead.

A retrospective meeting was also held once a week, which should contain the current problems and suggestions for improvement, but also positive aspects of the project. That helped the project flow and the satisfaction of the participants.

Open Source Software

The soTired project emerged as a university project in which all participants decided to implement an open source solution. This means that both, so the client application code and the server code are publicly available under the license * insert license here *. This means that everyone can view the code, edit it and use it for their own purposes if it can still be made available as an open source solution. This has the advantage that the study participants can see the code and which data is used and stored. This ensures data security. In addition, the code used here is made available to the science and can be further developed. Since the study project is very limited in time, there are some areas of the app that can be added and adapted later on. This can increase the quality of the code in some way.

Architecture

In this part of the report, the architecture of the application is described in more detail.

Architecture overview

Overall, the application is divided into two completely independent part. On the one hand, there is the stand-alone client app, which contains the self-test and a questionnaire. On the other hand there is the server application that gives an export functionality with the user data.

Stand-alone client application The client app includes the self-test, which is supposed to give reliable data about the test person several times a day after it has been carried out. The *Psychomotor vigilance task* and *Spatial span task* are two tasks included in the self-test. There is also a questionnaire, which is supposed to record the current condition of the subject at longer intervals. This happens via a certain number of questions with four choices each. There should also be a notification for the self-test, which asks the user up to three times a day to also carry out the test. In addition, the current mood and current activity of the test person is saved before each self-test so that the data can also be reliably interpreted. Last but not least, the client application also offers a settings page with options for client-server communication and to control this in a targeted manner.

As the client application is written in *dart* with the *Flutter* framework you can create multiple cross platform apps out of a single codebase [6]. This means that the client side is not limited to a single platform or operating system and the code does not have to be extended for other end devices.

Server application // TODO

User, client and server communication

For the communication between the user, the client an the server, one can say that the user only communicates with the client app. The client app then forwards the data to the server at a suitable time. The first time the app is used it gets a config file from the server, which initially load the information for the user. This architecture has the advantage that the user does not necessarily have to rely on a stable internet connection when carrying out the self-test or the questionnaire, and the data can also be transmitted later. At the same time, the low level of data exchange also ensures that the user only receives relevant information to save on his own device. In addition, the user is always aked whether the data should really be transmitted to the server. In figure 1 you can see the communication between user, client and server in detail.

Database management

The soTired application is known divided into two areas, the stand-alone client side as well as the server side with options for evaluation and export. In order to make both areas independent of each other and also to ensure that the values

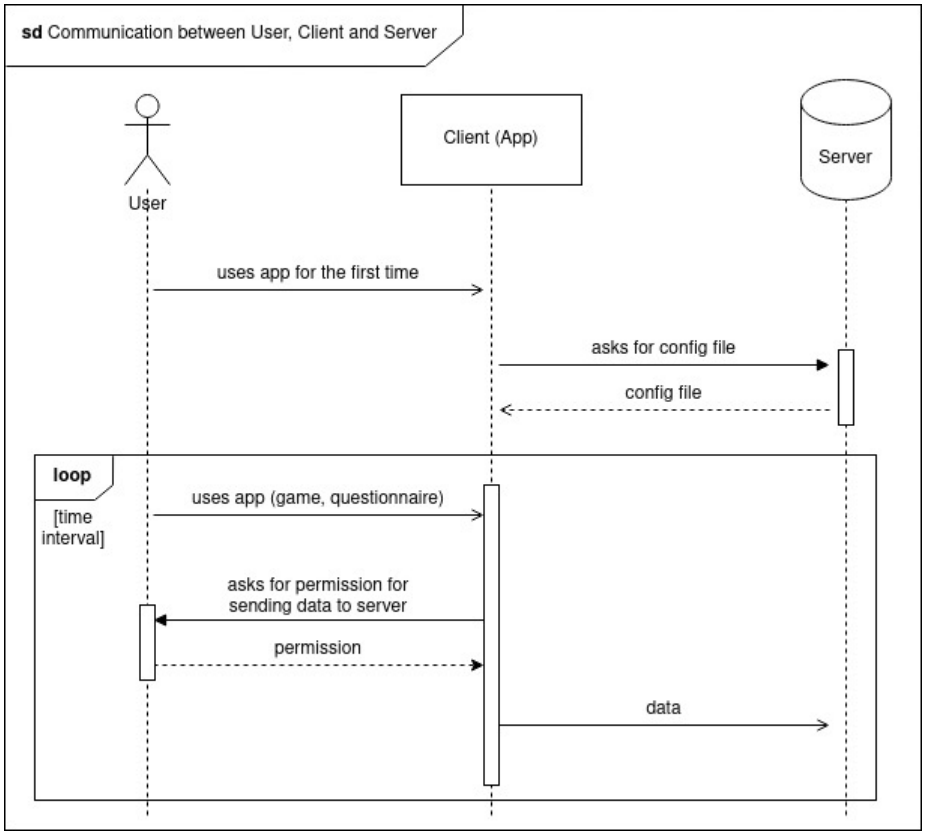

Figure 1: Communication between user, client and server

are always available, even if there is no network connection at a certain point of use, it was agreed to create and manage an independet database on both sides.

Client database

As described above, the client side is written with the helf of the cross platform framework Flutter and the programming language Dart. Hive describes itself as a 'Fast, Enjoyable & Secure NoSQL Database' [5]. That makes it easy to store database objects in boxes and access them again. Hive in conjunction with Flutter also promises great support in terms of use, as this has been specially adapted to it [5].

In addition, a NoSQL database was chosen because it is more flexible in terms of the variety of data in the area of research results and the speed required for storing such a large amount of data. Espically when thinking of adding additional functionality like audio or video recognition and using artificial intelligence to get information out of that. This is particularly suitable for this, as the data is only saved and not read out within the app. In this way, all data that is available can be collected without loss and no information is lost through a rigid relational form and can be sent to the server side.

Server database

In contrast to this, a relational database is used on the server side in order to be able to organize the results and export them later evaluated, if necessary. We decided on the lightweight, relational database SQlite that describes itself as an 'small, fast, self-contained, hight-reliability, full-featured, SQL database engine. This rather traditional, but well-known approach to storing large amounts of data promises above all a very structured distribution on the data, from wicht an export can later be made available via a .json file.

Overall, it can be said that the data structure on the server side looks like described in Figure 2. Therefore the main focus is on the study with the corresponding information and settings. The users assigned to a respective study are also stored in a separate table with the associated primary key for the study. This is where the assignment between user and study takes place. Then there are two tables that contain the log structure of the self test and the questionnaire. Inside the log tables you have all the information that need to be exported afterwards. At the same time, the questions of a specific study and the associated questionnaire are saved together with the selectable answers so that they can be accessed in the study.

User interface

The user interface in the app should be kept as simple as possible in order to be accessible to everyone. That means there is a start page with the logo and four options. From there, the settings lead to a subpage that provides certain

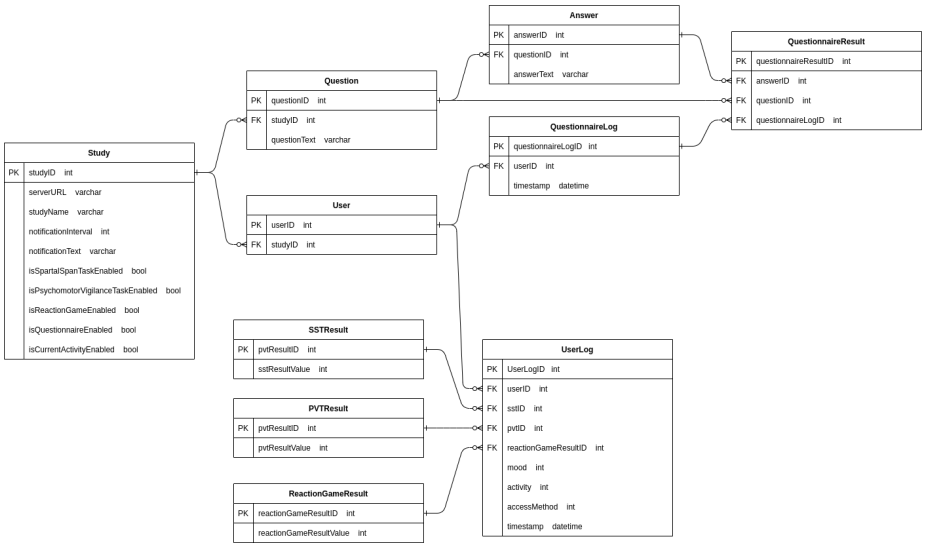

Figure 2: ER diagram

functions for the client-server function. The study URL can be selected here and data transfer can be ordered (Figure 3). There is also the sub-item of audio detection, which, however, has not yet been implemented due to time constraints and the limitations of the project. The two points above are the self-test and the questionnaire, which are illustrated in more detail below.

As described in Figure 4, the self test first depicts the current mood and activity of the test person by asking the test person 2 questions. To illustrate this, emojis have been used that should be known to everyone. The PVT test starts immediately after the self-assessment. As described in Chapter 2, this includes a test of reaction speed and attention span. Then the spatial span task starts (Figure 5). This in turn shows the respondent's attention span in which boxes that light up must be noted. It becomes more and more difficult depending on the level, as more boxes light up that the test person has to remember.

Some pop-up dialogs were used to display the current game status, which guide the player through the various tasks. In addition, all tests contain an overview of the current level and the score. This should give the user the feeling of knowing where you are in the game and how many tasks and areas are still to be expected.

Another feature of the app is the questionnaire, which starts immediately after clicking the button and can be answered intuitively. There is one question and four options for each question, which can be staggered. A progress widget is also available, which shows which question you are currently dealing with.

Overall, the user interface is kept in dark mode and adapted to smartphones, as the app was initially tested and improved on Android. Thanks to the cross-

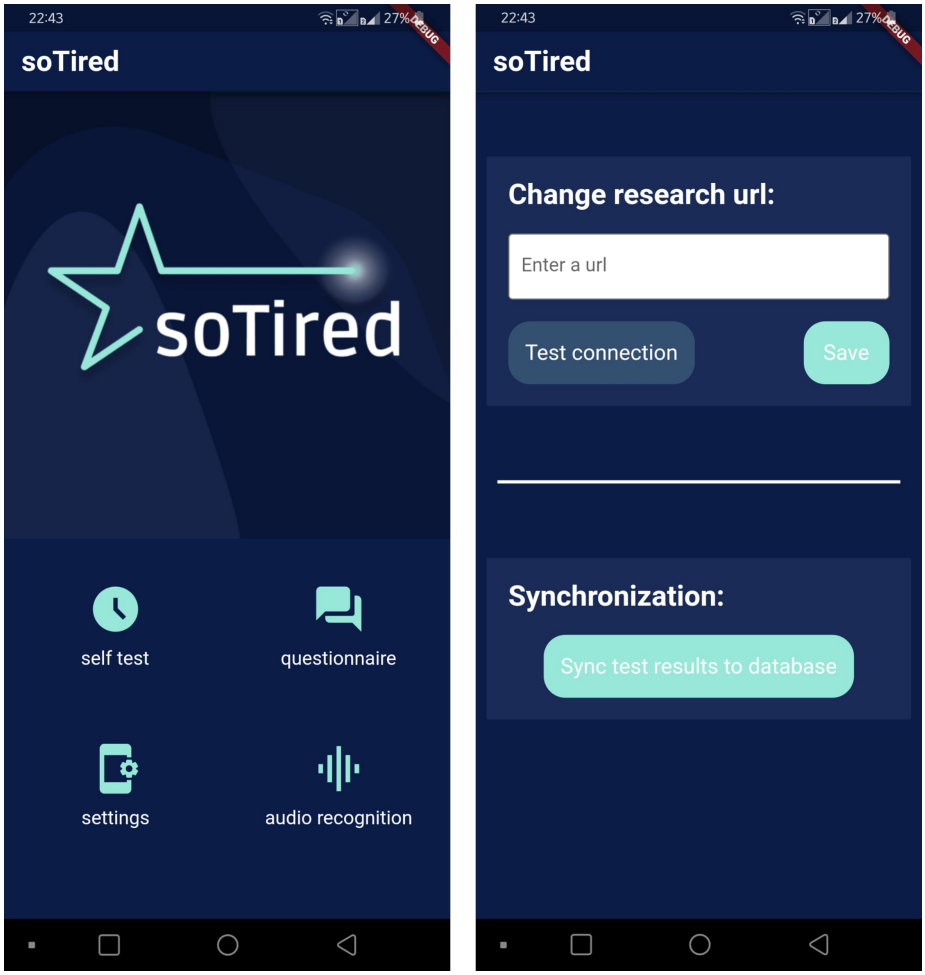

Figure 3: Home & Settings Page

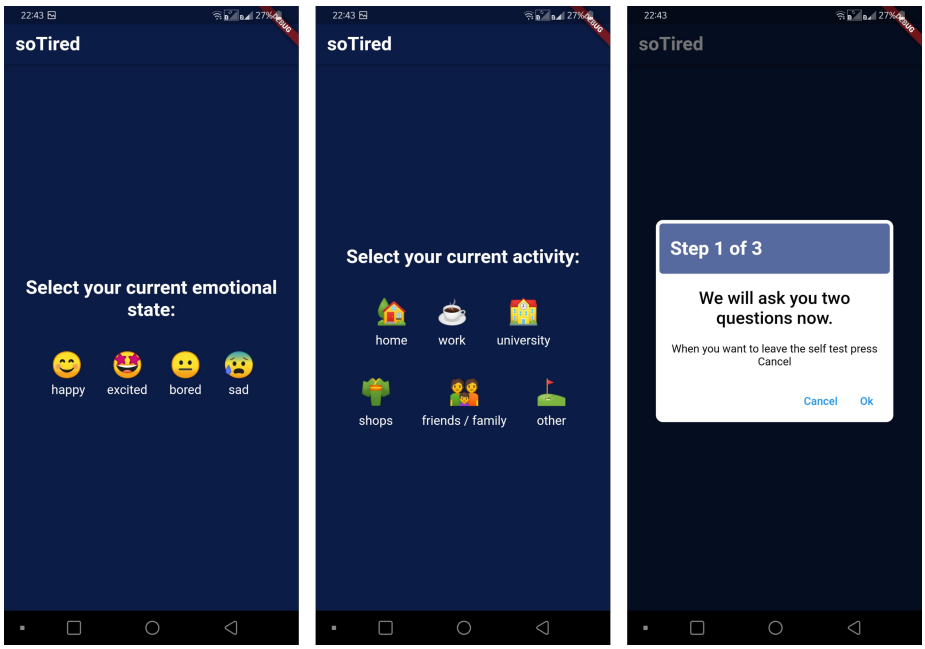

Figure 4: Self Assessment

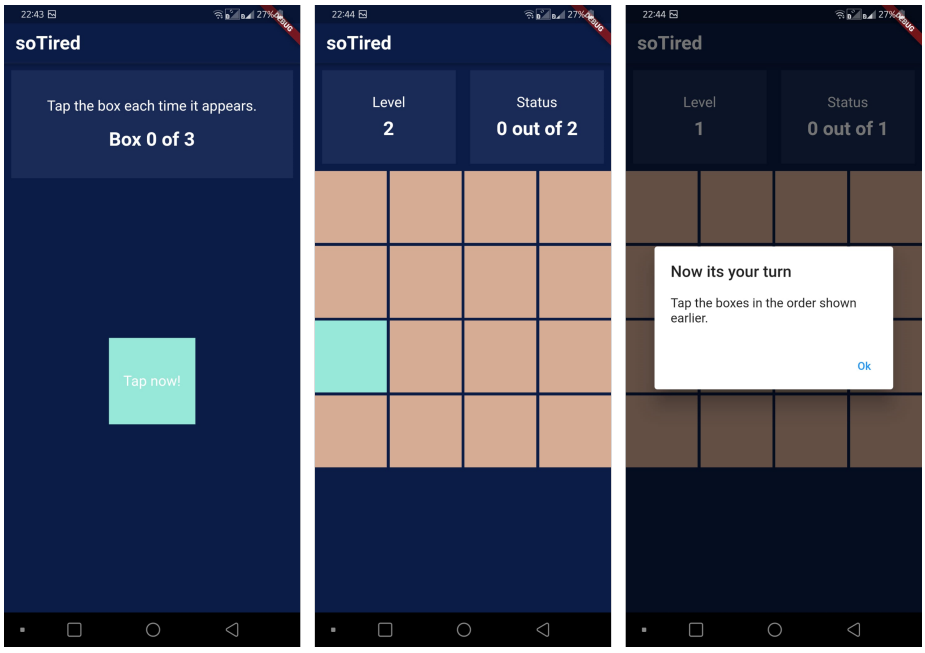

Figure 5: PVT and Spacial span task

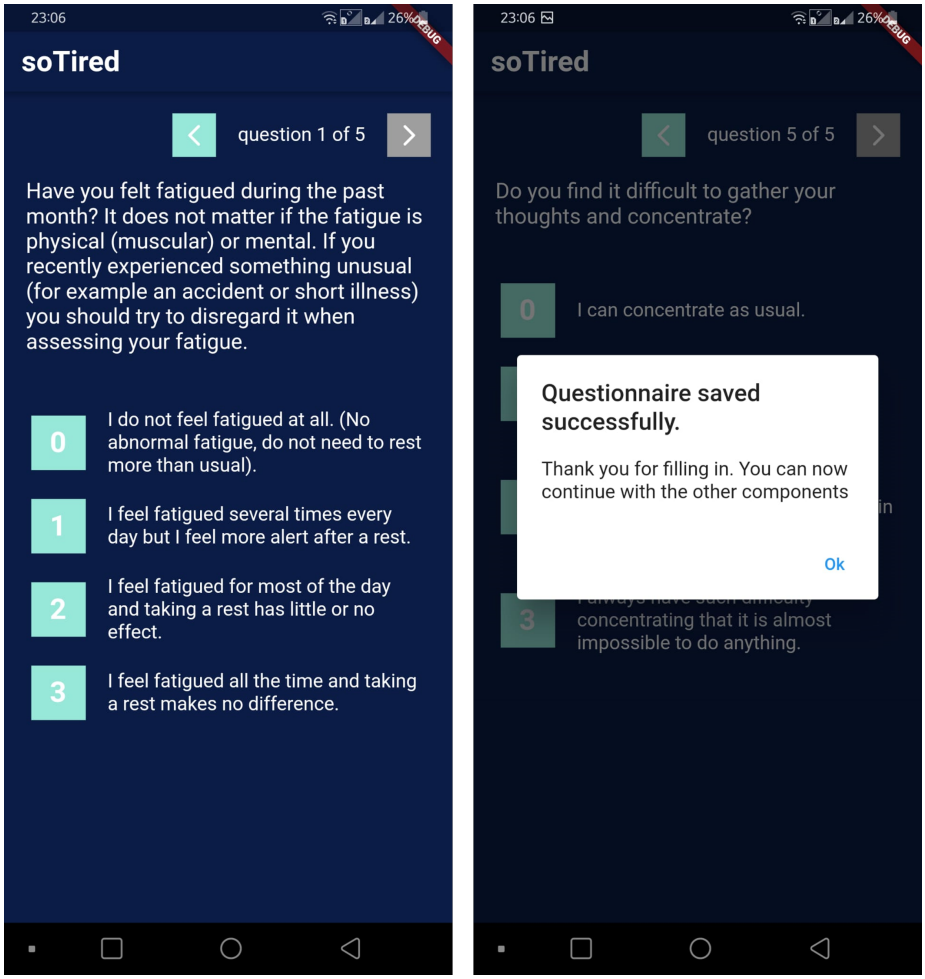

Figure 6: Questionnaire

platform accessibility, the app can also be accessed via the web and the areas can be controlled and edited. The color scheme is kept very subtle in pleasant shades of blue, the primary color is a bit detached with a light turquoise and is primarily intended to highlight things. The accent colors represent warmer tones and warning colors.

Client server communication

// TODO: Protokolle für die Kommunikation, was wird ausgetauscht? Format == JSON? Worauf muss geachtet werden? Braucht es Adapter / sonstige Besonderheiten?

Implementation

The implementation took place on the client side in Dart and Flutter and on the server side with golang. In the following, the reasons for the selection of the programming languages and frameworks as well as special features in the code are discussed.

Flutter application

Flutter is a open source framwork developed by Google. With Flutter you can create several applications (iOS, Android, web application) with just one code base. So the development does not take place platform-specifically, but in one programming language and environment.

The programming language used with Flutter is DART. DART ist a object oriented programming language that is very similar to C# or Java.

// TODO: Mehr Informationen zu DART / Flutter

All in all we tried to set up a reusable folder and file structure with some seperation of concerns between the frontend / user interface and the business logic.

Config

Database

Exceptions

Notifications

User Interface

The user interface (ui) part of the code is divided in a core, models and modules folder.

The core folder contains all files that have to do with the home page and navigation bar. So everything that has to be reached from everywhere inside the application.

The modules folder containes all seperate parts that act independently. So there is a folder and widget for the questionnaire, the spatial span task, the pvt test and the settings page for example.

Each module contains a dart file with the main Scaffold or Container that contains all the contents of the specific subpage. Then you have a widget folder with all seperate smaller parts of the application. So there is for example a QuestionnaireAnswer widget that is used for each answer of a specific question.

This precise separation ensures, on the one hand, that the code is much clearer and should also be less redundant. This avoids repetitions. In addition, the areas of responsibility are clearly separated from each other so that errors or changes only have to be carried out once and not several times. In addition, the individual areas should be easy to find by other programmers.

Go backend

sample text here.

Results and discussion

Sample headline 1

Sample Text

Sample headline 2

Sample Text

Conclusion

Sample headline 1

Sample Text

Sample headline 2

Sample Text

References

[1] Shortz, A.E., Pickens, A., Zheng, Q. et al. The effect of cognitive fatigue on prefrontal cortex correlates of neuromuscular fatigue in older women.

J NeuroEngineering Rehabil 12, 115 (2015). https://doi-org.ezproxy.hs-augsburg.de/10.1186/s12984-015-0108-3
[2] Wylie, G.R., Genova, H.M., DeLuca, J. et al. The relationship between outcome prediction and cognitive fatigue: A convergence of paradigms. Cogn Affect Behav Neurosci 17, 838–849 (2017). https://doi-org.ezproxy.hs-augsburg.de/10.3758/s13415-017-0515-y
[3] Scott, E., Feeling irritated, stressed, and finding it hard to get stuff done? You might have cognitive fatigue (2019). https://metro.co.uk/2019/02/09/feeling-irritated-stressed-finding-hard-get-stuff-done-might-cognitive-fatigue-8469750/
[4] Price E., Moore G., Galway L., Linden M. (2016) From Paper to Play - Design and Validation of a Smartphone Based Cognitive Fatigue Assessment Application. In: García C., Caballero-Gil P., Burmester M., Quesada-Arencibia A. (eds) Ubiquitous Computing and Ambient Intelligence. UCAmI 2016. Lecture Notes in Computer Science, vol 10069. Springer, Cham. https://doi-org.ezproxy.hs-augsburg.de/10.1007/978-3-319-48746-5_33
[5] hivedb.dev (2021) hive 2.0.4 https://pub.dev/packages/hive
[6] Flutter (2021) https://flutter.dev/?gclid=Cj0KCQjwnJaKBhDgARIsAHmvz6d__QztQXZcctcUwsX5U02GNS
[7] Nada, M.M., Maher, E.A., Basheer, M.A. et al. Can electrophysiological tests be used as screening tools in detection of cognitive impairment in obstructive sleep apnea hypopnea syndrome?. Egypt J Neurol Psychiatry Neurosurg 56, 25 (2020). https://doi-org.ezproxy.hs-augsburg.de/10.1186/s41983-020-00163-6 [8] Mathias Basner, MD, PhD, MSc, David F. Dinges, PhD, Maximizing Sensitivity of the Psychomotor Vigilance Test (PVT) to Sleep Loss, Sleep, Volume 34, Issue 5, 1 May 2011, Pages 581–591, https://doi.org/10.1093/sleep/34.5.581 [9] Cambridge Brain Sciences https://www.cambridgebrainsciences.com/science/tasks/spatial-span [10] Tokarev A, Flutter - der Hype aus der Sicht eines Nicht-Entwicklers (2019) https://www.new-communication.de/neues/detail/flutter-der-hype-aus-der-sicht-eines-nicht-entwicklers/