

TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Department of Software Science

Ain-Joonas Toose 204298IAPM

**A NOVEL DATA COLLECTION SOLUTION THROUGH  
DIGITAL FINE MOTOR SKILL ASSESSMENT FOR FATIGUE  
VISUALIZATION AND ANALYTICS**

Master's Thesis

**Supervisor**

Elli Valla

MSc

**Co-supervisor**

Sven Nõmm

PhD

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Tarkvarateaduse instituut

Ain-Joonas Toose 204298IAPM

**UUDNE ANDMETE KOGUMISE LAHENDUS  
DIGITAALSETE PEENMOTOORSETE TESTIDE KAUDU  
VÄSIMUSE VISUALISEERIMISEKS JA ANALÜÜSIKS**

Magistritöö

**Juhendaja**

Elli Valla

MSc

**Kaasjuhendaja**

Sven Nõmm

PhD

Tallinn 2022

## **Author's declaration of originality**

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Ain-Joonas Toose

.....

(signature)

Date: December 3, 2022

# Acknowledgment

The researcher would like to thank Elli Valla for their references to machine learning models, general expertise and grand amounts of help during the development of the application and analysis of data. As the original subject provider and recruiter to the research, Valla have proven themselves as a great mentor and researcher in the realm of fatigue testing. The researcher would also like to thank Sven Nõmm and Aaro Toomela for creating the basis' of the implemented test concepts and supporting the development.

# Annotatsioon

Väsimus on termin, millega saab kirjeldada inimese tööjõu ning aktiivsuse taset kui ka tema vaimse aktiivsuse suutlikkust. Parkinsoni tõve uurimisel on leitud mitmeid seoseid vaimse väsimuse ning neurodegeneratiivsete haiguste vahel, kus sümptomite esildumist on võimalik uurida samade lahendite alusel.

Töö peamine eesmärk oli leida lahendus väsimuse testimiseks, kus testis osalejal oleks võimalik anda oma tulem ajast ja kohast sõltumatult lihtsustades sellega uuringu läbiviimise protsessi. Töö skoobiks osutus mobiilirakendus Android operatsioonisüsteemil, andmete vaatluskeskkond veebirakendusena ning andmete valideerimine statistilise väärtuse hindamiseks.

Rakendus tõestas end efektiivse andmete kogumise viisina. Klassiruumides konkureeriva lahendusega 45-minutilisel testimise sessioonil leiti mobiilirakendusel kuni 30 uut testi lahendust. Eelnevaga koguti sama aja sees viis. Andmete statistiline väärtus valideeriti hüpoteeside tõestamise ning korrelatsioonide uurimise kaudu. Komplekt masinõppe mudelitest treeniti andmestiku peal, mis suutis eristada väsinud ning mitte väsinud isikute vahel.

Lõputöö on kirjutatud inglise keeles ning sisaldab teskti 57 leheküljel. Kokku on töös 9 peatükki, 23 joonist ning 10 tabelit.

# Abstract

Fatigue is a term that can be used to describe the spent capacity for mental, physical, or emotional workloads. During research into symptoms for neurodegenerative diseases, correlations have been found between mental fatigue effects and Parkinson's disease. As a primary goal for this research, the task of creating more data for research on fatigue was created.

The primary goal of this thesis was to build a framework for data collection. An application was built for the Android operation system, with an accompanying web application for data review and statistical analysis to confirm the value of data produced from the mobile application. A database was created with a REST API to serve data from it to provide a single source of data.

The application provided a more efficient method of data collection. In a time frame of 45 minutes, five instances of a competing test could be performed while up to 30 results could be made with the mobile application in the same period. A set of hypotheses were used to prove data validity, a linear association assessment was performed, and a set of machine learning models were implemented which could distinguish between fatigued and non-fatigued participants.

The thesis was written in English at 57 pages, including nine chapters, 23 figures, and 10 tables.

# List of abbreviations and terms

API	Application Programming Interface
ASD	Archimedes spiral drawing test
CRUD	Create, Read, Update, Delete
DOM	Domain Object Model
JPEG	Joint Photographic Experts Group, image file format
JSON	JavaScript Object Notation
KMP	Kotlin Multiplatform
MVC	Model-view-controller
SPA	Single Page Application
SSR	Server Side Rendering
RT	Reaction Test
RTA	Reaction Test, Advanced
RTS	Reaction Test, Simple

# Table of Contents

<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	3
1.2 Structure . . . . .	3
<b>2 Overview of technologies</b>	<b>5</b>
2.1 Kotlin . . . . .	5
2.2 Typescript . . . . .	6
2.3 Kotlin Multiplatform . . . . .	6
2.4 Spring, Spring Boot . . . . .	7
2.5 Svelte . . . . .	8
<b>3 Solution architectures</b>	<b>11</b>
3.1 Mobile application . . . . .	11
3.2 REST API . . . . .	14
3.3 Data Dashboard . . . . .	15
3.4 Solution Architecture . . . . .	16
<b>4 Mobile Application</b>	<b>22</b>
4.1 Terms and Conditions, Metadata . . . . .	23
4.2 Reaction Test . . . . .	26
4.3 Spiral Drawing Test . . . . .	29
4.4 Tremor Test . . . . .	31
4.5 Upload . . . . .	31
4.6 Tutorial . . . . .	32
<b>5 REST API</b>	<b>34</b>
<b>6 Data Dashboard</b>	<b>36</b>
6.1 All Tests View . . . . .	36
6.2 Test Details View . . . . .	38
6.3 All Devices View . . . . .	40
6.4 Device Tests Details View . . . . .	41



<b>7</b>	<b>Assessing Statistical Value of Data</b>	<b>42</b>
7.1	Experimental Setting . . . . .	42
7.1.1	Feature Engineering (ASD test) . . . . .	43
7.2	Hypothesis Control . . . . .	44
7.3	Hypothesis Control Results . . . . .	46
7.4	Linear Association Assessment . . . . .	48
7.5	Machine Learning . . . . .	50
<b>8</b>	<b>Discussion</b>	<b>54</b>
<b>9</b>	<b>Summary</b>	<b>56</b>
	<b>Bibliography</b>	<b>58</b>
	<b>Appendices</b>	<b>62</b>
	<b>Appendix 1 - Non-exclusive licence for reproduction and publication of a graduation thesis</b>	<b>62</b>
	<b>Appendix 2 - Terms of Service PDF provided to test taker before being allowed to start tests</b>	<b>63</b>
	<b>Appendix 3 - PDF of research request sent to secondary education schools in Tallinn.</b>	<b>69</b>

## List of Figures

1	Generated sequence diagram of CrudController’s create() function. . . . .	18
2	Database graph of tests tables and relations. . . . .	19
3	Sequence diagram of the uploading to REST API process. . . . .	20
4	Sequence diagram of dashboard communication with the REST API for table-based views. . . . .	20
5	Sequence diagram of dashboard communication with the REST API for detail views. . . . .	21
6	Screen view of terms and conditions. . . . .	24
7	Screen view of the metadata collection fragment. . . . .	26
8	Screen view of the simple aiming reaction test, the starting and progress date are shown on the left and finishing state shown on the right. . . . .	27
9	Screen view of the advanced aiming reaction test. First image shows situation where colors do not match. Second shows the test situation where test taker should interact. Third shows the finished state. . . . .	28
10	Image of the ideal Archimedes Spiral rendered as the template. . . . .	30
11	Screen view of the ASD test in an initial and finished state. . . . .	30
12	Screen view of the Tremor test. . . . .	31
13	Screen view of upload progress and success view. . . . .	32
14	The user interface of the tutorials for each test. . . . .	33
15	Browser capture of the initial view and main device table view. . . . .	37
16	Mobile device view of content presented in Figure 15. . . . .	37
17	Browser capture of the test metadata and simple reaction test visualization. . . . .	38
18	Browser capture of the ASD test painted by a test taker and rendered in the dashboard. . . . .	38
19	Browser capture of tremor test visualizations. Above shows velocity for x,y,z axis. Below shows absolute acceleration. . . . .	39
20	Browser capture of the device-based overview list view. . . . .	40
21	Browser capture of the test metadata and tab view of the device-based test details view. . . . .	41
22	Visual representation of the differential-type (a) and angular-type (b) features . . . . .	43
23	Image of Spearman correlations between ASD tests classifiers and numeric groupings observed by metadata. . . . .	53

## List of Tables

1	Table of day and night Welch and Mann Whitney U-Test p-values. Values under alpha are presented in <b>bold</b> . . . . .	46
2	Table of self-assessed fatigue grouped Welch and Mann Whitney U-Test p-values. Values under alpha are presented in <b>bold</b> . . . . .	47
3	Table of mental work grouped Welch and Mann Whitney U-Test p-values. Values under value are presented in <b>bold</b> . . . . .	47
4	Table of physical work grouped Welch and Mann Whitney U-Test p-values. Values under value are presented in <b>bold</b> . . . . .	48
5	Table of Spearman correlations between simple reaction test times and numeric groupings observed by metadata. . . . .	49
6	Table of Spearman correlations between advanced reaction test times and numeric groupings observed by metadata. . . . .	49
7	Table of Spearman correlations between tremor tests for right-hand absolute acceleration and numeric groupings observed by metadata. . . . .	49
8	Table of Spearman correlations between tremor tests for absolute left-hand acceleration and numeric groupings observed by metadata. . . . .	50
9	Fatigue classification cross-validated model performance. The best scores for each test are presented in <b>bold</b> . . . . .	51
10	Best performing machine learning models for fatigue classification. . . . .	52

# 1. Introduction

At the time of writing the article, fatigue has been a topic weighing heavily on people as a state of consciousness that can sometimes be hard to describe. By anecdotal experience, the scientific discussion is not singular. When discussed in the day-to-day debate, questions about fatigue appear with answers vaguer than expected. While humans can determine physical fatigue by the changes in their muscle response [1] there are forms of fatigue that are more difficult to quantify into a set of predictors. Concepts like tiredness [2] or loss of focus [3] come up when talked about mental fatigue. Emotional fatigue can be described as fatigue towards reacting to outside stimuli or frustration [4]. While muscle fatigue has determined classifiers one can pinpoint without intrusive methods (spasming, shaking) [1], the two others mentioned have many different tests, of which most are either differential or require the use of eye measuring with specialized cameras [5], or measuring blood oxygen levels [6].

While there have been previous tests developed for spiral draw assessment[7] and the mobile market holds reaction test based fatigue assessment methods, most research has been found to focus on extremes with fatigue assessment, often falling back to using expensive eye-tracking software or creating closed test conditions where the test taker, supposedly taking a mobile test, finds themselves in the same conditions of control any extremely precise test would put them in. In the theme of data collection, the researcher, therefore could find themselves in a situation where they have a resource in potential data sources in the form of test takers, but not available to perform these tests in a way the researcher nor test taker would find it acceptable in terms of time spent, effort taken or validity. Therefore, if a fatigue assessment dataset can be collected and found viable by user input and self-assessed data, this could provide much in research on fatigue assessment by fine motor ability. As there exists previous proof about spiral drawing tests and their ability to assess mental fatigue [7][8][9][10] if the spiral drawing test taken in the application under this research can provide similar results, an issue of spiral drawing test result dataset size would find a valuable solution.

A common complaint with spiral drawing test research is the lack or small amount of data available. Be it Dzotsenidze approaching the issue of data generation of generative data augmentation networks to produce more variants of spirals [11], Senkiv finding lack of

data as a possible cause of not finding muscular fatigue results in the test [8] or findings by Valla and Nõmm that taking spiral drawing tests even with parties that are not affected by a neurodegenerative disease like Parkinson's as time-consuming or performance hindering. Because of current methods of approach, this research and application developed expect to produce a method of data collection that could be used by any person having access to an Android-based newer smartphone and a dataset and immediate review method for anybody with access to test and research upon said findings.

There has been previous work on both spiral drawing tests and fatigue assessment. Still, no research has been yet done using an Archimedes spiral drawing test incorporated into an easily accessible mobile application where the test giver would not have to be at the exact physical location as the test taker. Nor has a fatigue test application built with specific questionnaires and assessment methods. Furthermore, current researchers of the spiral drawing test and its affinity for use have yet to have a database with the breadth of data available. A dashboard to quickly visualize it nor to implement it into a pipeline that could be updated after every test taken should they so want.

To present a method for data collection, a battery of tests had to be assembled with data from every test available for immediate review. While there have been mobile applications developed by Tallinn University of Technology before, the test collection proved challenging to perform because of the closed conditions of it - the test had to be done in person, using a specific tablet with the software specific to the model without public availability. While providing good results, the amount of data collected proved to be insufficient for future work, whilst data collection for machine learning model creation was to be the goal. For many neurodegenerative diseases, the common symptom of mental fatigue has been proven[12]. Still, model accuracy, besides providing a potential fatigue assessment method, could be extended by data from people whose medical history concerning neurodegenerative disease has not been proven.

During the document, multiple references can be found to the previous work done by Senkiv and Nõmm, Toomela regarding the performance of spiral drawing tests regarding mental fatigue in numerous frames of reference. While Senkiv built machine learning models that could differentiate mental fatigue states using kinematic parameters[8] Nõmm and Valla have both presented methods and models for Parkinson's disease classification using the spiral drawing test and its multiples of kinematic features extracted from them[10][9]. Reaction time tests have seen high use in fatigue mapping before with Migliaccio's work using reaction time tests on athletes with aim-based tests[13] to small-sided games, and reaction tests measured using Stroop test by Coutinho [14].

## 1.1 Problem Statement

The goal of this thesis was to produce a solution to data scarcity with respect to fatigue tests and to provide a more easily accessible method of testing. To evaluate the quality of data received, the data acquired using the software solution would have to be assessed in terms of statistical significance. In all, three goals would be achieved and compose the problem statement:

- Solve the issue of data scarcity in two ways:
  - Provide an accessible setting for test subjects to participate in research.
  - Build a dataset structure and database to store it.
- Verify if the proposed solution would decrease test-taking efforts in terms of time and resources spent.
- Answer the question of such an approach having a value in data quality.

A decision was taken to write a mobile application to assess fatigue. A collection of tests were assembled. Two modifications of an aiming-based reaction test are given together with a spiral drawing test and a tremor test to assess hand movement while attempting to stay still. After these tests, data was gathered and assembled in JSON format and saved to a Postgres database. As the expectation was to create a test that was simple in execution and rich in collected data, a questionnaire was devised with decisions on a scale taken from previous research.

## 1.2 Structure

This work describes the decisions taken and technologies used for application development, highlights the architectural decisions taken during development, and produces an overview of 3 applications: the mobile application used for data collection, the REST API and database used to store test results, and data dashboard created for reviewing collected data. An assessment of the statistical value of the dataset will be provided by analysis already made using the achieved datasets with hypothesis testing, linear association assessment, and a machine learning model built on ASD tests. In all, the work will be presented in 2 parts:

- Development of the solution. An overview of technologies, solution architecture, and developed application overview will be provided in sections two to six.
- Assessment of the statistical value of data will be provided in section seven, describing the experimental setting, feature engineering, hypothesis control, linear

association assessment, and machine learning models developed and used.

## 2. Overview of technologies

As the amount of development required for a software suite is extensive, as much was attempted to be used in frameworks as possible. This chapter describes the larger frameworks used in each. A short introduction to each language and framework used will be given, with a paragraph explaining the decision to use them added. In the case of web framework selection, multiple were selected for evaluation as the breadth of available frameworks is wide, and the decision required a deeper understanding of each approaches advantages and disadvantages.

### 2.1 Kotlin

Android development for native applications is usually done in either the programming languages Java or Kotlin. While Java has been described and used exhaustively in research and enterprise applications, Kotlin is still relatively new. First released in 2012 [15] Kotlin has quickly risen to the contention of languages employing the Java Virtual Machine (JVM) by the quality of components and interoperability with previous Java code. A Java project can take on Kotlin code without any extra configuration needed[16] and immediately make use of the features Kotlin provides like:

- Null safety
- Data classes
- Coroutines
- Performant custom control structures
- Well-working string templating
- Class access by property
- Singletons

and more[15]. As the primary supported language of Android, the decision to use Kotlin for as much of the codebase as possible was taken.



## **2.2 Typescript**

Javascript was developed as a small scripting language with the purpose of adding trivial interactivity to websites. Still, it has become a language used for server software in the case of NodeJS, full application development loops, or even native applications. While the complexity of the language has grown exponentially, some of its original design choices have remained. The affect of development time and complexity are out of scope of this work.

A common mistake that ties down development time when assembling browser-rendered websites is ignoring type mutation. One of the main accusations laid at Javascript has been type-safety - comparing a number to a string is an allowed pattern, an object can be described as a string, number, an object, or an array at any moment. The language has no in-built safeties to circumvent that compile-time.

Typescript has been defined as a static type checker for Javascript, a transpiler language. The main goal of the language is to assign types to javascript statics and objects and to ensure that types would not be mutated during their lifecycle. This allows a developer to write Javascript code in a strongly typed manner [17].

The decision of Typescript was taken in the interest of data dashboard development ease. With previous experience using the language, the researcher found using the extended toolset Typescript provides on objects to quickly find possible points of application failure and ensure these would not happen. In a similar manner to the Kotlin decision, Typescript was decided for use because of extended use by the programming community, previous experience, and the language's qualities to hasten the development of the data dashboard.

## **2.3 Kotlin Multiplatform**

One of the initial goals was to create an application that could be easily extended to platforms other than Android. As the previous solution had shown difficulty when popular platforms change, the technical description of the project included initial support for iOS and Android.

While unified application technologies exist and have been extensively used in the form of piped web browser instances and Flutter applications, the necessity of data collection and the use of paint features necessitated the use of system native code. Previous experience of the researchers had proven that using methods of approach where loading native code

through wrappers or other methods typically proved the application to either become more difficult than necessary in terms of development or stifled expected results.

Since timing was important for RTs and device position data with high accuracy important for tremor tests, the goal for used technology was to allow as much native code as possible but integrate user interface elements. For this, only one possible technology was available.

Kotlin Multiplatform (KMP) is a technology designed to simplify the development of cross-platform projects [18]. With KMP, a unified framework for much of the user interface components of the application could be built once and extended to both IOS and Android operation systems. For parts where native code was to be implemented, the addition worked seamlessly. Allowing the developer to write hybrid code in Kotlin, Android native code in Kotlin, and iOS native code in Swift, KMP was chosen because of these qualities. While the framework is still in its beta stage, the application could be quickly split into different native applications should there be a necessity.

## **2.4 Spring, Spring Boot**

The Spring Framework provides a comprehensive programming and configuration model for modern JVM language based enterprise-level applications. A key element of Spring is infrastructural support at the application level, allowing the developer to focus on the content of objective code creation and removing the need for implementing levels of application development that can be replaced with generics. The framework allows the developer to use code collections named Beans to integrate support for SQL processing, data mapping, security, and others, thus removing the need to implement code that otherwise would have to be built when creating a CRUD application. [19]

Spring applications are typically defined by an MVC design pattern where models are commonly defined by the response from the database and handled by domain mapping and specially designed and named repository objects. The bulk of the processing is typically done on the level of other bean-assisted objects called services. Often in cases of pure data representation, another object type, mappers, is relied upon to translate database responses to their appropriate form. Another object, the controller, works as the mediator of request, the start of processing, and the response to the mapped API call. In the case of the application used, the backend, the HTML queries are used by either the frontend application or the Python pipeline.

The decision to use Spring came from the researchers' prior experience writing applications that correspond to an MVC pattern with Spring and its support of Kotlin in a native manner.

Making use of the framework sped up the development of the backend and allowed the research application to not become slowed down by custom model approaches or controller design implementations. Having become an enterprise standard of software development, Spring has shown itself as a powerful framework, making use of dynamic database approaches when loading test data as a JSON object from the database. Using included patterns, defining search patterns and paging for optimization becomes a task of assigning flags or annotations on object classes rather than extensive software development exercises.

## 2.5 Svelte

As is the case with using native Java for implementing an MVC pattern, implementing a user interface without the help of language extensions and supersets of code provided by a framework slows down any extended development. A data dashboard requires at least a basic implementation of a CRUD workflow. Though in this case a read-only approach was selected, the reading of data in a visually simple manner still requires the use of tables, search functions, paging support, routing, and other functionalities.

While all parts can be accomplished without using a framework, implementing even a paginated table in a performant matter can take more effort than expected. Javascript as a popular language for web applications has hundreds of frameworks available with a selection of popular frameworks. During the planning of technologies for use, attempts were made to start a project with React, Vue, Angular, and Svelte. The initial expectations while testing out the frameworks were:

- Ease of routing
- Ability to quickly transform data from API response to a visual component
- Overall speed of loading the page
- Amount of "boilerplate" or code required for the setup of a new page required

React, being the most popular of selection, requires the developer to immediately start deciding on approaches for data saving across sites ("Stores"), and how to perform an API call as multiple competitors exist. This sounds true for almost all of the parts of the framework. As an open framework, thousands of extensions have been built on to React, where some attempt complete control of the development experience (Next.JS) and others only over, per example, saving data over route changes (Redux, MobX). A developer will have to spend valuable time developing code to connect all their development decision results together and can create a codebase of more than 10 classes before a simple table with data from the backend application is even rendered. React can be a natural choice

for many when building a single-page application and is an excellent choice for large codebases, but can be difficult in the setup phase of the application. Furthermore, because any package decided upon also has to exist in the NodeJS ecosystem and contains its own set of packages, the size of a project can increase exponentially, and the same for rendering and compilation time. The researcher has years of experience with the framework and could still find themselves working more on setup than business logic.

Angular is another popular choice. As the framework built by Google, the design language reflects that of a standard Android application when set up. While the framework has more decisions made for the developer when originally set up, this translates to a large amount of boilerplate the developer has to know how to modify to not run into compilation issues. As the researcher was experienced with the framework, this process was not foreign to them, but the method of piping API response results to the template and the way how the template is built, loading asynchronous data objects, which count for almost all objects in the data dashboard, becomes an exercise in predicting site render times.

While the researcher was foreign to Vue, an attempt was also made to create the application using it. Vue is a progressive framework that allows one to quickly start development [20]. Many of the steps expected from a React or an Angular application could be ignored and implementing a table could be done faster than the last two applications compared. Yet when dealing with the template and asynchronous calls, the development can slow down due to lack of experience as the templating system does not follow the style an HTML site would expect them to. While Vue attempts to provide a template that looks like HTML, much of the implementation mutates the approach.

The last framework and the one chosen for final development, was Svelte. What defines the difference Svelte from the others attempted, was the compilation process. A Svelte site does not start an instance of the site where the state is dictated by a virtual DOM built by the framework but rather attempts to manipulate the DOM presented by the browser, retaining much control for the developer to develop or debug on the DOM level[21].

The added benefit of such an approach is also an increase in the speed during site load. Much of the user interactions that are developed in a Svelte UI have been compiled before the built code is loaded and rendered by a browser. The templating and development process in Svelte proved also to be fast. Of the attempted solutions for decision-making, getting a website with a table and detail route took the least time. Nearly no boilerplate code was necessary as SvelteKit, an accompanying framework provided by Svelte, has a simple routing system that leans into using folder structures as the route-building method. Transforming data from an API call empowers javascript's own inbuilt methods without an

intermediary. Page load speed, though not tested with much depth, was the fastest. As this filled all criteria the best, Svelte was chosen for the development of the data dashboard.

Though the selection of a web framework is often something done by preferring previous experience while setting up the project, in a quickly changing ecosystem like the one around Javascript, keeping an understanding of current developments is important. Svelte marks a step towards developer comfort in the researchers' opinion and will be used by them again when building projects the size and scale of the data dashboard again.

## 3. Solution architectures

In this chapter a general overview of the patterns and approaches to the development of the mobile application, backend and frontend will be given. The general flow and structure will be provided and a description of reasonings behind them. The overview will not describe design patterns like dependency injection or facade patterns as these describe general approaches to software development. Barring from these constructs, an explanation of flows and a description of the separation of views will be given.

### 3.1 Mobile application

The application is comprised by Views. This means that for every page the application holds, a class extending the android View class is built. These are loaded by a class named Intent, which handles the rendering of the view and timing for it. Every view in the application is held by an Intent, managed by the main class. As such, the main class becomes a glue for state management, database interactions and authorization. While every View exist with an Intent behind it, the main tests Intent is called in another abstraction layer named Fragments. The Android FragmentStateAdapter is a class that extends the RecyclerView.Adapter class and is used in conjunction with the ViewPager2 widget to manage and display Fragments in a swipeable, horizontal layout. The FragmentStateAdapter is responsible for creating and managing the Fragments that are displayed in the ViewPager2, and it maintains the state of each Fragment as it is added, removed, or updated. The ViewPager2 widget uses the FragmentStateAdapter to populate the pages that are displayed, and it provides the necessary scrolling and gesture support for users to swipe between Fragments. This allows developers to easily create swipeable layouts with a consistent and efficient user experience.

A Fragment in Android is a modular section of an activity that typically has a distinct user interface and logic. It is a reusable component that can be added to an activity to create a multi-pane user interface, allowing developers to create more complex and dynamic layouts. Fragments can be used to display new content in the same activity, or to display the same content in a different way in separate activities. They can also be nested inside other Fragments to create more complex and hierarchical layouts. Fragments have their own lifecycle and can be added, removed, or replaced within an activity without affecting

the rest of the activity's UI. This allows developers to create more modular and flexible user interfaces.

These Fragments and their lifecycle is controlled by a tab based layout. For that, the class extension of TabLayout is used. Each Fragment represents the content of a different tab, and the TabLayout widget is used to display the tabs and allow users to switch between them. The Fragments are added to the TabLayout using an adapter, such as the FragmentPagerAdapter or FragmentStateAdapter, which manages the Fragments and their corresponding tab titles. When a user clicks on a tab, the corresponding Fragment is displayed in the activity's content area.

The TabLayout widget can be used with a ViewPager to provide swipeable tabs, or it can be used on its own to create a static tabbed interface. The TabLayout widget in Android is a horizontal strip that is used to display multiple tabs with different content. It is typically used in conjunction with a ViewPager to provide swipeable tabs, or it can be used on its own to create a static tabbed interface. The TabLayout widget is typically placed above the content area, and each tab is represented by a tab title and an optional icon. When a user clicks on a tab, the corresponding content is displayed in the activity's content area. The TabLayout widget can be customized with different colors, fonts, and styles to match the overall app design. It also supports features such as tab selection indicators, tab scrolling, and tab swiping.

In this instance, the lifecycle of tabs is controlled by a button that appears on the bottom of every Fragment when the test conditions have been matched. The tabs are not interactable in the application and serve only to keep focus on current test step and to allow the user see what steps have been finished and which ones are left.

Some Views are though called out with an Intent. The application holds some Views that are controlled by it. For example, the application holds a terms and conditions check that provides a link to a PDF the user has to agree to before starting the test. If an user has not accepted the terms, the application will close immediately. If the user accepts the terms and conditions, the state of it will be saved in the application's local storage and will stay so until the application has been updated. When the application version were to change or the user re-installed the application, that state would be nulled. To control this, the main class will attempt a check for the locally stored variable and start the Intent if either the entry has not been found or previously declined. Another example of such a process exists for Tutorials. Tutorials are checked by another locally stored variable, where the tutorial will always be shown on first entry and after that, on every third visit as a reminder. The intent checks for these are stored for every Fragment, where the Fragment, before

hydrating the View it is called to, will determine if a tutorial Intent should be loaded.

As mentioned, while Fragments can start Views, they can also check for Intents and load new Views before starting or during a View. Views themselves can load new Intents. An example of this exists in the Tremor test. While the Fragment-loaded View of the tremor test holds 2 buttons and a tab layout lifecycle control button, the buttons load begin an Intent on which a View load is done - directing the user to the tremor test for the specific type of the 2 tests.

Modern Android local database handling is typically done using the Room persistence library. This library provides an abstract layer over SQLite, allowing developers to create and manage a local SQLite database using simple Java objects. The Room library provides a powerful and flexible way to create and manage databases, allowing developers to define the database schema using Java or Kotlin classes and annotations, and to query and update the database using an API. The Room library also supports features such as database migration and transaction management. The local database holds information on previous test results made on the device and saves test results for each test after the user has begun the process of moving to the next test. The database also holds the generated ID identifying the device. To ensure uniqueness of results, an UUID is generated every time a test session is started. That UUID will be attached to each test and will become the filter by which test results would be polled.

When all tests have been completed, the data gets sent to the backend application. On the final view, all test results will be collected from the database pertaining to that specific instance which is controlled by the UUID. An interface exists for these tests called a TestHolder. From that, a generic is formed which holds data every test produces and another collection of JSON for test-specific data. REST (Representational State Transfer) can be handled in Android using the HttpClient library. This library provides a low-level API for making HTTP requests and managing the response data. To use HttpClient, developers must create an HttpClient object and configure it with the desired settings, such as the request method, the request headers, and the request body. They can then use the HttpClient object to make the HTTP request, and to parse and manipulate the response data. Using Kotlin's coroutines, an asynchronous process is made to POST that query result.

In Android, the runBlocking function is a suspending function that is used to run a coroutine in a blocking manner. This means that the calling thread will be blocked until the coroutine completes, and any other coroutines that are waiting for the result of the blocked coroutine will also be blocked. The runBlocking function is typically used to run a coroutine in a test



environment, or to run a long-running operation in the background while the main thread continues to run. It is important to use the `runBlocking` function with caution, as blocking the main thread can cause the application to become unresponsive and can lead to poor performance.

## 3.2 REST API

The backend application is built using a MVC structure. MVC (Model-View-Controller) is a design pattern that separates an application into three distinct components: the model, the view, and the controller. The model represents the data and the business logic of the application, and it is responsible for storing and manipulating the data. The view represents the user interface of the application, and it is responsible for displaying the data and providing user input. The controller acts as the intermediary between the model and the view, and it is responsible for coordinating the interactions between them. The MVC pattern allows developers to create more modular and maintainable applications by separating the different components and their responsibilities. It also allows for easier testing and debugging, as the different components can be tested and modified independently.

In the application, most classes have their core logic moved to a Generic. Generics are a feature of the Kotlin and Java programming language that allows developers to specify the type of data that a class or function can operate on. This allows developers to create more flexible and reusable code, as they can define a class or function once and use it with multiple types of data. In a Spring Kotlin application, generics are often used with the Repository interface, which provides a generic way to access and manipulate data in a database. For example, a developer might define a Repository class that can operate on any type of data, and then use generics to specify the type of data that the Repository will be used with. This allows the Repository class to be used with different types of data without having to create separate classes for each type. In this instance, also Service and Controller has a generic structure, where for most Services, there lacks a need to even describe a class. As Controllers are described by an annotation, most of controller logic is also handled by generics.

As an example for dataset save functionality, the controller for `EntityHolderController` extends `CrudController`, `EntityHolderService` extends `CrudService` and `EntityHolderRepository` extends `CrudRepository`. In the case of a save functionality, the variations of `TestHolder` begin from the controller endpoint, find object name and relation by descriptors, and continue to a service function that does validations - raising an exception should there be an ID added already since an update case misses the purpose of the POST endpoint created. After validating, a call to the repository save method is constructed. The create

endpoint is simplistic, but describing it once as a generic and extending it by creating a class and describing an endpoint and entity name makes it able to describe the whole save process for all models in the application through a single sequence diagram shown in 1.

While the logic of services is mostly solved with generics, the domain objects are separated into multiples. While a central controller exists named TestHolder, another also exists with the name of TestHolderBasic. This is due to the fact that a test can hold more than 5000 lines of JSON and always querying them while requesting a list can be computationally expensive enough to cause slowdowns in browser rendering time. The TestHolder structure employs the same generic structure initiated in the mobile application. The TestHolderBasic structure omits the test-specific data extension.

The object design described in the mobile application is replicated in the PostgreSQL database. Though every test has their table, the existence of them in that manner is mostly to control table size and database readability. As such, for every test type there exists a table that shares shape. To collect all the tests into a DTO structure, a separate table exists which creates simple foreign key relations with the other tables. The tests employ the JSONB structure for the test specific JSON data. JSONB is a binary representation of JSON data that is optimized for efficient storage and manipulation in a database. JSONB data types allow developers to store and query complex, hierarchical data structures in a PostgreSQL database.

The backend employs Spring Data JPA for repository logic and pagination. It is a library that is used to define and execute query specifications in a Spring-based application. Providing a simple and intuitive API for defining query criteria, Spring Data JPA allows developers to create complex and dynamic queries without having to write in low-level query languages. Spring Specification API is used to provide a search functionality. It provides a simple and intuitive API for defining query criteria and allows developers to create complex and dynamic queries without having to write SQL or other low-level query languages.[19]

The backend works as a RESTful API where no PUT, PATCH or other endpoints than GET and POST exist. GETs are only available for use by the data dashboard and POSTs for the mobile application.

### **3.3 Data Dashboard**

While Svelte can be ran in a method that is browser-rendered and act as a traditional site, the approach chosen for the dashboard took use of using a NodeJS server to run the

application. Although creating a Single Page Application(SPA) out of it functionally, the application is still rendered on the side of the browser. This means that an initial site load could be slower than an application rendered on the side of the server, the overall performance of the site will be better because of a lack of extra steps taken for rendering.

A reactive site is a web application that uses the principles of reactive programming to create a responsive and interactive user experience. Reactive programming is a programming paradigm that is based on the principles of data flow and event-driven programming. In a reactive site, the application is designed to react to changes in the data and the user interactions, and to update the user interface in real-time. This allows the application to be more responsive and resilient to changes, and it makes the user experience more fluid and seamless. A reactive site typically uses reactive streams, observables, and subscriptions to manage the data flow and the user interactions, and to update the user interface in real-time.

The data dashboard is a RESTful, reactive application. Only the endpoints it employs are the GET endpoints from the backend which it does by paginated queries. As the backend provides search functionality and pagination support, the dashboard uses asynchronous processes and listeners to poll data on search or page change events. This allows to pull data at only, in this case, 20 objects at a time. Most of the reactive processes come from Svelte's inbuilt template method named `async` since the complexity of queries warranted no need for other solutions. The template functionality provides a `waiting`, allowing the site to wait on content rendering in the specific part of the site until data is available.

The dashboard is comprised of 4 views:

- A table view that provides all test holders.
- A detail view that shows a specific test holder's content and provides an overview with visualizations of the data.
- A device table showing all devices which have participated in the testing.
- A device dashboard, which loads an instance of the detail view for every test selected in that view.

### **3.4 Solution Architecture**

While each application of the solution holds its own purpose, the benefit of a process created can only be realized when all parts are communicating with each other - the mobile application without a service to save data into would provide no data and a data dashboard with no data to consume would be useless. All main parts of the application have communication between parts. Even the analysis pipeline has access to the data,

though that is solved by direct database access to simplify data consumption.

All applications communicate with the REST API. The tasks of the REST API have been split by the purpose of the applications connecting to it. A read-only dashboard and write-only mobile application provide a single source of truth for data by the database for which the REST API provides the access to . Without a way to modify it, data quality is ensured by not providing any PUT or PATCH endpoints.

The lifecycle of a test is described by the process of saving the performed test in the mobile application, the save of data to the backend application and the read of data by the data dashboard. Both steps can be described by simple sequence diagrams as a largely single-step process since the communication models between the application require no more. For the mobile application, the POST command returns the created object.

For the dashboard, the communication starts from loading into the site. As the backend endpoint provides two separate GET endpoints for data provision - a GET endpoint for a single test with the parameter of the requested test ID as input and an endpoint for multiple tests that holds a search functionality for filtering. In the REST API, the find process is described in figure 4.

The communication model for the GET processes can be described as a sequence diagram described by the two different endpoints. An example is provided for the find relation. As with the previous figures, describing the solution infers a simpler model and does not describe all the steps.

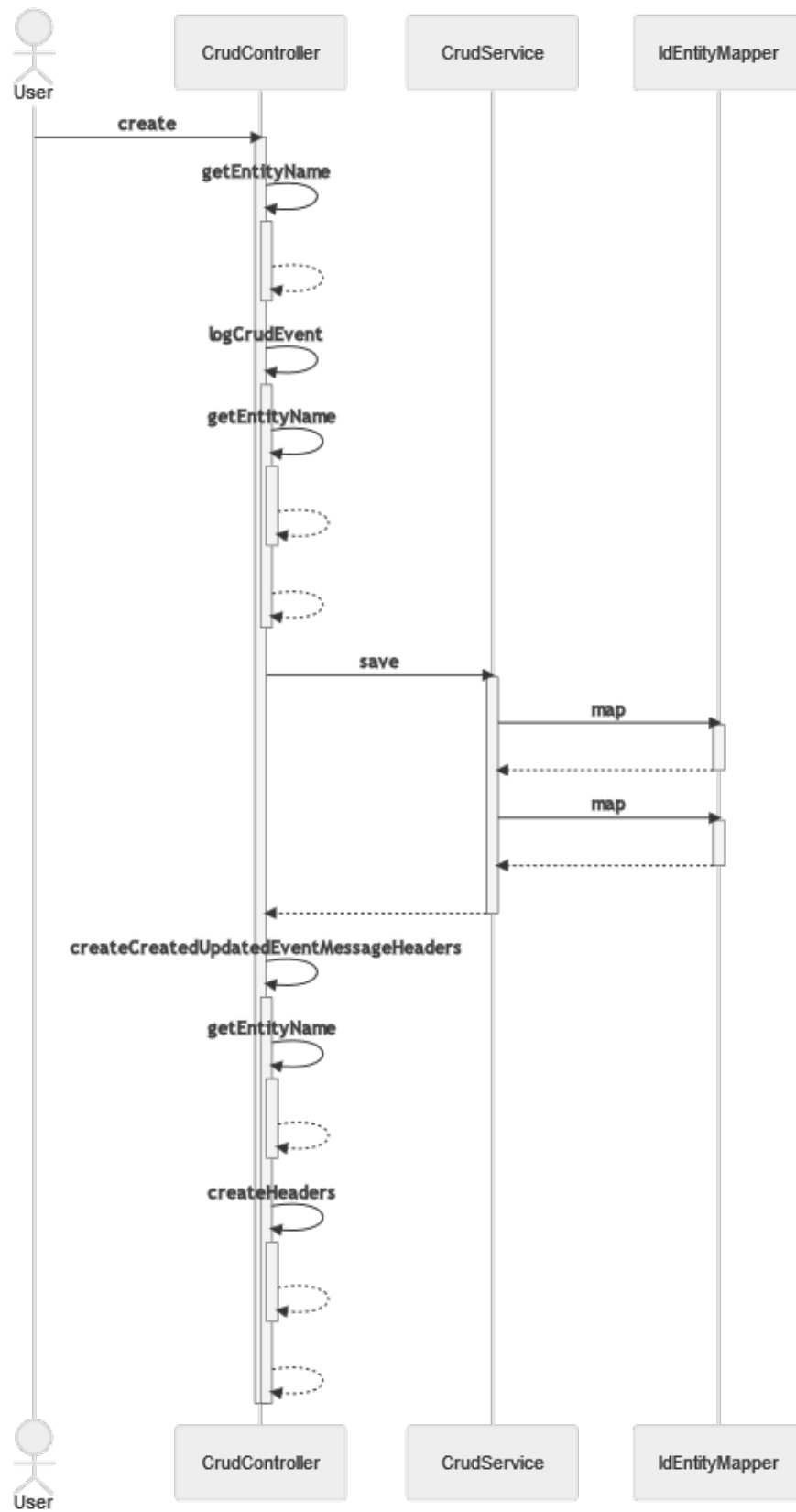


Figure 1. Generated sequence diagram of CrudController's create() function.

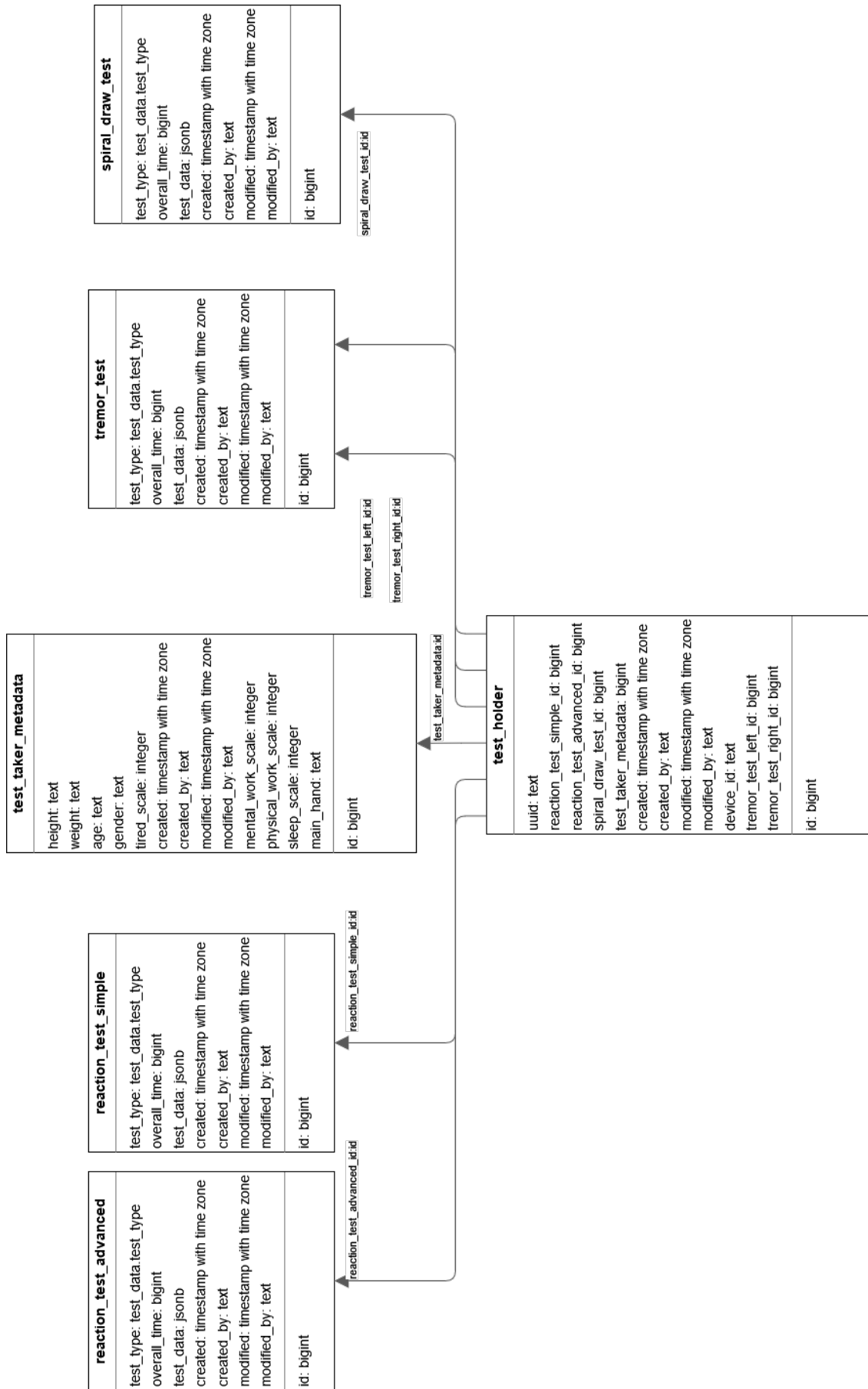


Figure 2. Database graph of tests tables and relations.

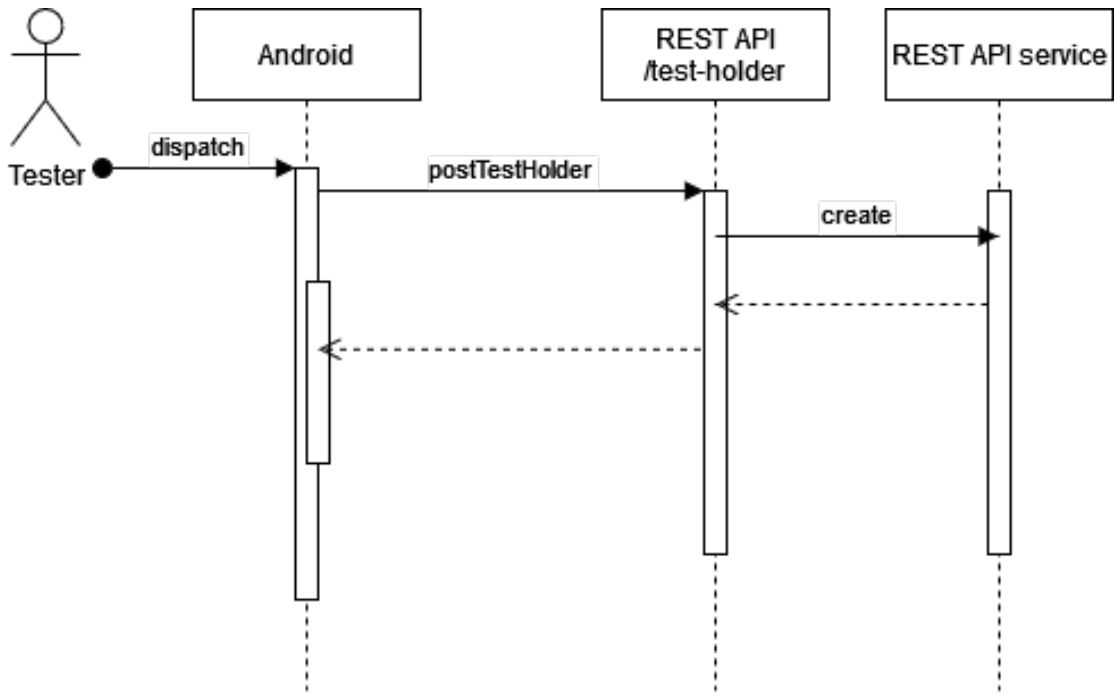


Figure 3. Sequence diagram of the uploading to REST API process.

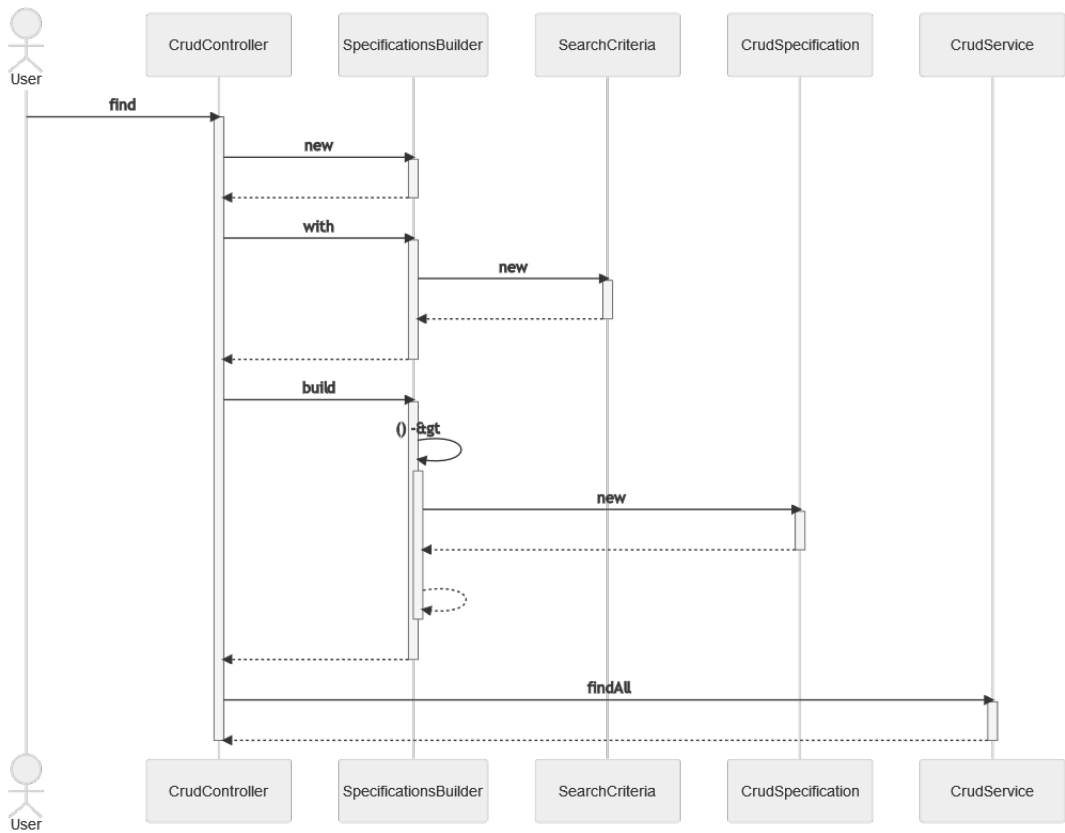


Figure 4. Sequence diagram of dashboard communication with the REST API for table-based views.

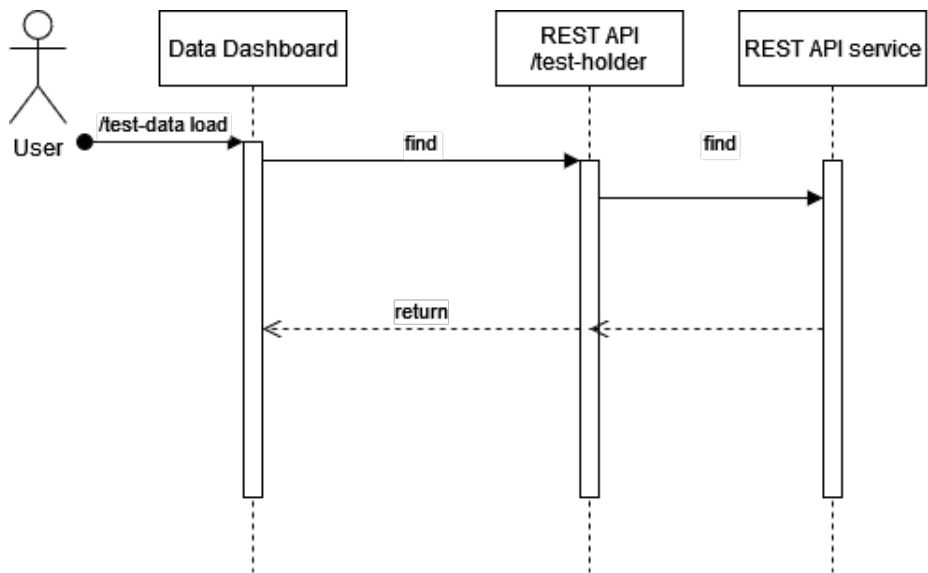


Figure 5. Sequence diagram of dashboard communication with the REST API for detail views.



## 4. Mobile Application

The main goal of this research was to create a mobile application to provide a collection of tests with data collection accurate enough to use in current and future research on mental and muscular fatigue. With the further goal of analyzing patients with neurodegenerative diseases like Parkinson's or mild cognitive disorders whose symptoms have been proven to include mental or muscular fatigue, the data has to be proven as accurate and results gathered from it provable of having statistical value. With this, some issues can emerge. Accuracy can be affected, besides conditions, the application asks, by the user's stance, position while taking the test, attention to the test, or even random movements that could not be excluded. Because one of the goals is to provide this method of taking tests without a spectating party, some base decisions had to be taken. Before any development could begin, initial expectations were set, those being:

- Tests must not expect the use of any device other than the user's personal smartphone. This expectation excludes blood pressure analysis tools, smartwatches, or eye trackers that must not be used for the test.
- The tests have to be easy for the user to understand. With tutorials at every step of the test, visual helpers, and an easily seen way of understanding the current state of progression, the application had to be used for users without help from a third party. While the test could be used for collecting results after small-sided games or other encapsulating test scenarios, the application must not dictate the terms.
- The application must collect accurate and usable data. Any method of data collection should be set to the maximal levels of precision, and test conditions and tutorials must accurately describe the state of the process of the test.
- While anonymous, the researcher must be able to easily assess which test belonged to which device, through which assessment of user state to performance could be followed and analyzed
- The tests must be performed as tests assessing fine motor ability. Any test following movements like running could not be used because of it.

The conditions set removed much of the fatigue assessment methods currently in use. While Hu and Lodewijks have found success in tracking eye movement to assess mental fatigue through an extensive camera system [5], their tests were not focused on fine motor skill

nor could they provide test scenarios accessible for most people - the tests were performed on transportation industry members such as airplane pilots or professional drivers [5] with hardware mounted to the drivers' area. Recent small-sided game tests using commercial heart rate monitors have also shown success. Coutinho et al. found success in recording soccer games and assessing a measurable change in mental fatigue measured by Stroop test results before and after the football game[14]. Migliaccio performed tests using a digital variant of the Stroop test while also using heart rate monitors to follow athlete mental fatigue before and after various athletic tests[22]. While they presented a possible path to take with both the Stroop test and a device, these were ruled out due to a device measuring heartbeat that could connect to the application would not be available, all that could be otherwise applicable to taking the test. Using the Stroop test could have been used, but it also became ruled out due to the length of the test performed.

Research succeeded in determining previous connections between mental fatigue and spiral drawing tests. Senkiv had proven the previous applicability of a spiral drawing test performed on a tablet computer to have repeatable patterns for different assessed levels of mental fatigue.[8]. Migliaccio had found a difference in subjects' reaction speed before and after inducing both mental and muscular fatigue [13]. Lippold described connections between muscular fatigue to hand tremors in a closed position test[23], Budini assessed there not to be a difference in hand tremors for mental fatigue[24]. Still finding a good path to assessing muscular fatigue, the tremor test became the last test to form the collection of 3 types and a total of 4 different tests that became the base of the application. With a short description of the metadata collection scheme, terms and conditions gathering process, and upload, this forms the mobile application.

The test intends to be easily accessible, promoting additional attempts. A test taker is motivated to attempt at least twice - once in the morning and once in the evening with optimal results available after seven days of daily test taking.

The mobile application was realized as a Kotlin Multiplatform application with a release for Android devices from API version 26 to API version 33. It is available for all to download in Estonia at the time of research. As a section on the development techniques and architecture exists separately, this will expand on the tests, decisions taken with them, and data collected from their performance of them.

## **4.1 Terms and Conditions, Metadata**

Performing any test using people requires them to be aware of how the data is processed and the level of specificity of their person. The contents of the data, collection scheme,

and usage were presented at the first step of the assignment. Before using the application, the test taker has to accept the terms and conditions of the application. This is resolved by providing access to the document, hosted as a .pdf file. The tester will be redirected to a browser instance. After navigating back to the application, the user can decide whether they agree to the terms or not by clicking a button in the interface. Pressing "ACCEPT" allows the test taker to continue the test-taking process. Pressing "DECLINE" will immediately close the application. The terms of use document have been added as Appendix 2.

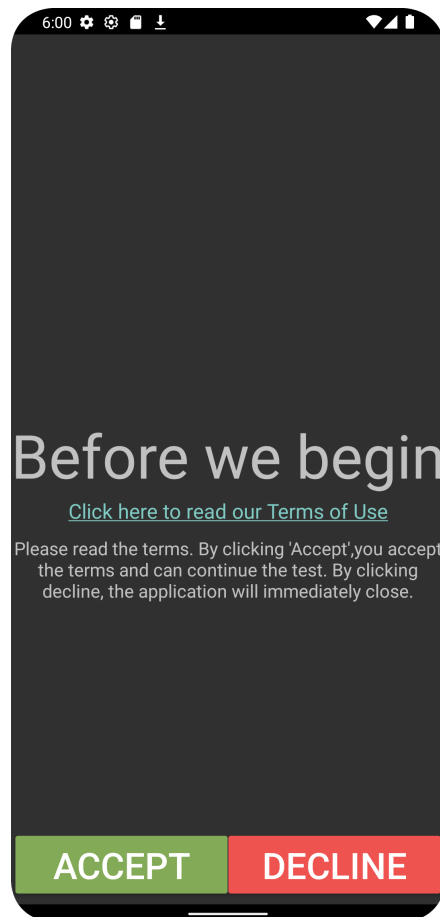


Figure 6. Screen view of terms and conditions.

Before taking a test, the user must also provide the following non-personally identifiable information (non-PII) information:

- Age in provided range descriptions: under 18, 18-25, 26-30, 31-35 were selected as the ranges to target most probably test taker age range. This can easily be extended.
- Height in centimeters in provided range descriptions: Under 100, 101 - 150, 151 - 175, 176 - 185, 186 - 190, 191 - 205, above 205
- Weight in kilograms in provided range descriptions: Under 50, 50 - 60, 61 - 75, 76 - 90, 91 - 105, 106 - 120, above 120
- Gender in provided range descriptions: female, male, other

- Dominant hand in provided range descriptions: left, right, ambidextrous
- Self assessed severity of tiredness in a scale of 1 - 10. Descriptors were provided for each option of tiredness:
  - 1 - No exhaustion
  - 2 - Very very slight
  - 3 - Very slight
  - 4 - Slight
  - 5 - Moderate
  - 6 - Somewhat severe
  - 7 - Severe
  - 8 - Very severe
  - 9 - Very very severe
  - 10 - Maximal
- Number of hours slept last night in the range of 0 - 12 where any above 12 is taken as 12 hours
- Number of hours of physical activity performed before taking the test from 0 to 12
- Number of hours spent doing any mental activity from 0 to 12

Following is a short literature review of existing tests and scores with their applicability to the one attempted, and decisions for exclusion:

- Fatigue Severity Scale(FSS) is a scale used in an academic context. A 9-item scale implementing a 7-point Likert scale, Krupp states the levels of severity by 1 for describing the highest self-assessed level of disagreement and 7 as the lowest while also describing 7 as the highest level of severity and 1 lowest level [25]. This seemed a possible candidate but fell out of contention because of the questions, not all being applicable to the mobile application's intent.
- Chalder Fatigue Scale was reviewed. Chalder Fatigue Scale is an extended version of FSS which is comprised of 11 questions pertaining to the same assessment qualities as FSS. This was ruled out for the same reasons as FSS was.
- Short Form Health Questionnaire(SF - 36) was considered. It is an overall assessment of a person's health with fatigue described under a Vitality scale. As Lins-Kusterer had examined the Vitality scale for determining fatigue for HIV patients[26], there seemed to be applicability. The questionnaire was ruled out because of the amount and scale of the questions. As SF-36 is performed as a boolean test[27], its classification properties of it were not sufficient.
- Checklist Individual Strength is a test with well-documented effectiveness. Worm-Smeitink found CIS to have high correlations to previously validated test results but noted a 17% false positive rate [28]. As a test consisting of 4 different chapters with

long self-assessment schemes, this also failed as a candidate for use.

After consulting with Aaro Toomela of Tallinn University School of Natural Sciences and Health, a possible user of the test in the future, the previously described questionnaire was compiled. Applying parts of FSS and using Likert scales where applicable, the test conditions were assembled. In the application, the metadata collection is done in a singular view with a mobile-friendly build, using sliders for Likert scale items and dropdowns for the range description options. The metadata page also displays the device ID, which is used later for determining device identity for test collection.

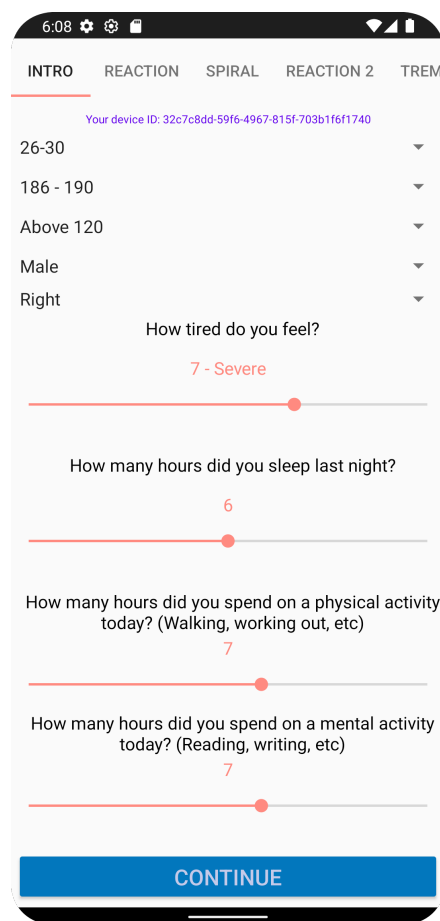


Figure 7. Screen view of the metadata collection fragment.

## 4.2 Reaction Test

The first and third test of the mobile application are two versions of a reaction test. In academic context, reaction tests have been found as an effective way of measuring mental fatigue. Multiple variations of Stroop tests have been performed with athletes after small sided games. With both mentions of Migliaccio and Coutinho in relation of the research [13] [14], reaction tests effectiveness can be proven as a valid assessment method in regards of determining mental fatigue. Stroop [29] described their test as a color to meaning test,

where an assortment of colors with separate, color describing, options were provided to them. The objective of the test was to describe the color of the shown color card to them as quickly as possible. Error rate and reaction time were recorded.

The Stroop test, while possible for implementation in a mobile application, was found to not work in a mobile application when following the parameters set for the application. A Stroop test is often performed in extensively long test periods - a 50 repetition Stroop test was found as minimal in all work mentioned before. A solution was found in modern aim trainer tests, typically focused for competitive e-sports players and people looking to enhance their aim in video games. The primary objectives of a Stroop test are to determine time from an input appearing to the reaction provided to it by hitting an input. An aim trainer works in a similar manner, though swapping the appearance of a color in any form for a target on a 2 dimensional plane the user treats as the plane for providing input. While adding some complexity, the basic principles of the test and data collected fit well to the parameters presented and proven over the past 87 years to the one Stroop described. An aim test translates well to a mobile application. With the added complexity of aiming, 15 attempts for reaction test were chosen.

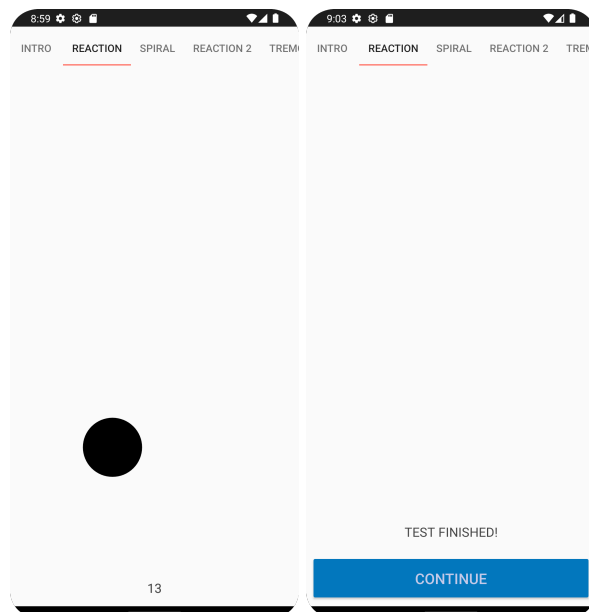


Figure 8. Screen view of the simple aiming reaction test, the starting and progress date are shown on the left and finishing state shown on the right.

The aim test comprises of 15 targets rendered after every time one of them is removed by pressing on one. The targets are rendered one by one at a random position on the screen in the applicable test area. The size of the target varies randomly between 45 to 150 pixels. The objective for the test taker is to click on the target as quickly as they can for 15 times. After the first target has been hit, data collection begins.

During the test, every input is recorded. If the input is in the same location as the target, a data point is saved with the recorded reaction time as time since end of rendering the target until touch and another point saved to identify the input event as a target hit. Should the test taker miss the input, the reaction time is still recorded though with an identifier to describe the input as a missed move. As per description, a minimum of 15 points are recorded. As the target does not render again in case of a miss, there isn't a limit set for missed hits. For user experience, a counter follows the current step of the test, counting down to 0.

The test is finished when the 15th target is hit. When that happens, the counter is replaced by a button, allowing the user to navigate to the next test.

Another variation of the same test exists. While an aiming test follows the principle of a reaction test, another version was implemented in the test suite. Taking the color matching logic from the Stroop test, this variation of the aiming test expects the user to only react to a target that shares the color with a color guide rendered every time a correct target has been hit. This version makes use of the user interface to provide a color guide and renders a new target every 1.75 seconds. The data gathered is similar, though another point of reaction time is saved through recording reaction time from correct color hit.

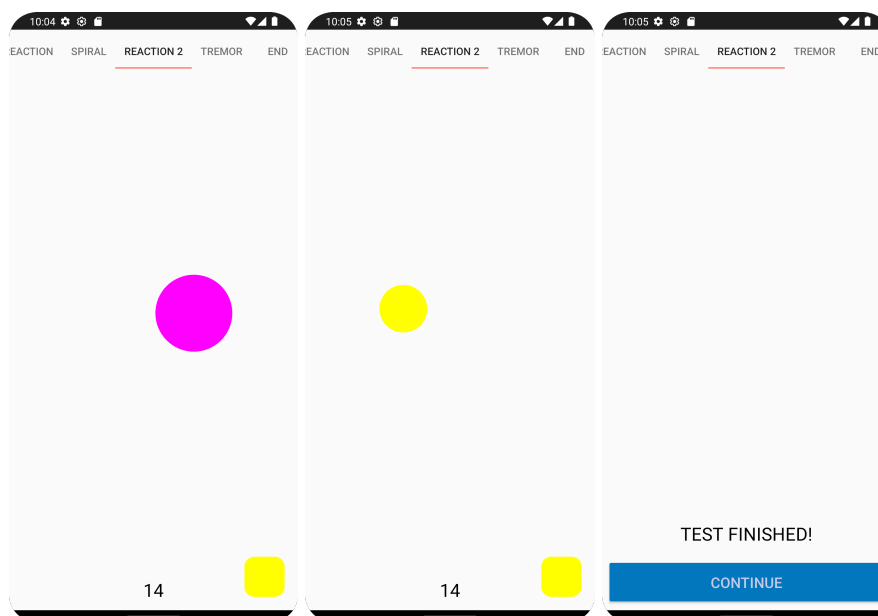


Figure 9. Screen view of the advanced aiming reaction test. First image shows situation where colors do not match. Second shows the test situation where test taker should interact. Third shows the finished state.

This variation also does not randomize colors indefinitely. 5 colors form a pool of randomization, of which a color not fitting the rendered guide color will leave from the seeding pool. This ensures that at the latest, the fifth color rendered will always be and continue

to be rendered with the expected color. The purpose of the added difficulty was to gauge an expanded level to the reaction test. Increasing the difficulty should produce a slower reaction time because of the added validation by the person and therefore differences in reaction time between different attempts should be more easily found.

### **4.3 Spiral Drawing Test**

The second test of the suite was selected to be the archimedean spiral drawing (ASD) test. Senkiv described the spiral drawing test as a previously popular and long-digitized test used among practitioners.[30] Drawing of the Archimedean spiral is a commonly used neurological diagnostic test technique to measure fine motor activity. Spiral analysis has long been classified as a clinically valid method for the objective evaluation of disorders like Parkinson's disease or tremor disorder. This means the methodology is valid for testing and definitely usable in the use case this suite is striving for.

In its analog version, the archimedean spiral drawing test is performed by use of pen and paper. The test subject demonstrated the template of a spiral drawn on the paper. To perform the test, the subject is asked to follow the contour of the spiral with the pen.

There are some variations of the ASD test that exist. A spiral can be drawn inside-out, outside-in, following the template by line, or following the whitespace between the lines. This test was the first one suggested originally for the development of the mobile application and the primary objective initially. While the rest of the tests came into development by Aaro Toomela, this was the test brought forward for the research by Elli Valla. The development of a version of the spiral drawing test fitting a smartphone-friendly form factor was the initial primary objective of the work. A previous digitized version of the test developed by Senkiv [8] was taken as the base for data collection and later feature extraction, though modifications were made because of the change in form factor from a pen and tablet-based approach to the one selected for this research.

The version of the spiral drawing test decided for the suite works as an outside-in test where the user would have to paint in the whitespace between the lines. This was decided because of the use case - the expected device for test-taking was described as a mobile smartphone. Without access to a smart pen or other similar device expected for all users, a test following the template would have produced an unnecessary amount of errors. With user tests describing frustration with seeing the line under their finger, a decision was taken to set the whitespace between lines as the medium of following.

The test starts with the user painting between the lines, starting from the right, and following



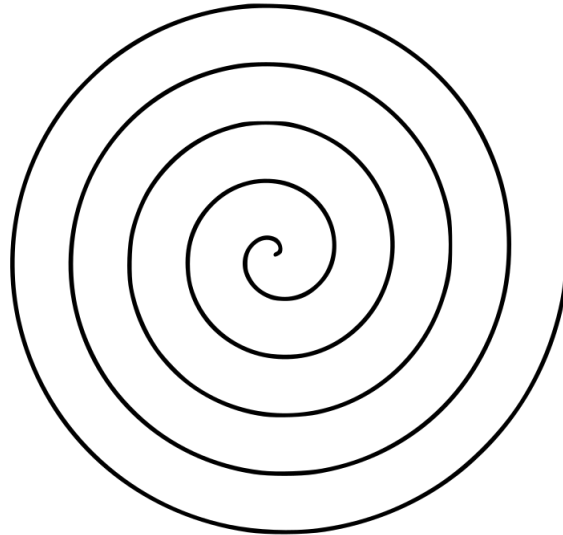


Figure 10. Image of the ideal Archimedes Spiral rendered as the template.

the lines until hitting the inside of the spiral. The test end is decided by the test taker as the "Continue" button will appear after the first touch has been done.

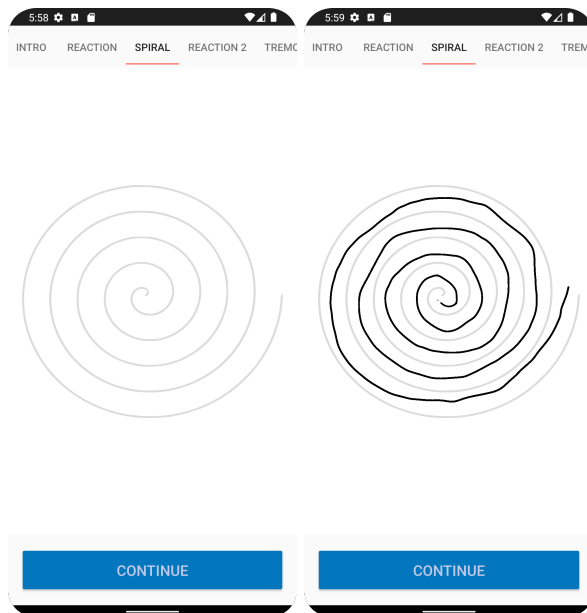


Figure 11. Screen view of the ASD test in an initial and finished state.

Data collected from the drawing is acquired every 15 - 17 milliseconds. A recording of the X and Y coordinates is taken with a reading of error hits and time between the last saved touch event. From this, a large selection of kinetic variables can be calculated, without the need for much more for the application to do while saving. As time is a premium with 17ms of time between touches, not much processing would be done during the test to keep the application performant.

## 4.4 Tremor Test

The final test of the suite is the tremor test. While Budini [24] found there not to be a connection to mental fatigue with hand tremors, discussions with Aaro provided a case for a test that could specifically target muscular fatigue. As a suite of tests, the reaction and spiral drawing tests were both proven beforehand to have an effect on mental fatigue, a test that could specifically target muscular fatigue was yet missing.

The setup for the tremor test is simple for the tester. The phone is to be held in front of the person, hand fully extended. With the phone facing the tester, it is expected of them to wait 10 seconds until the test is completed. A successfully taken test is accounted for when the tremor test has been passed using both hands.

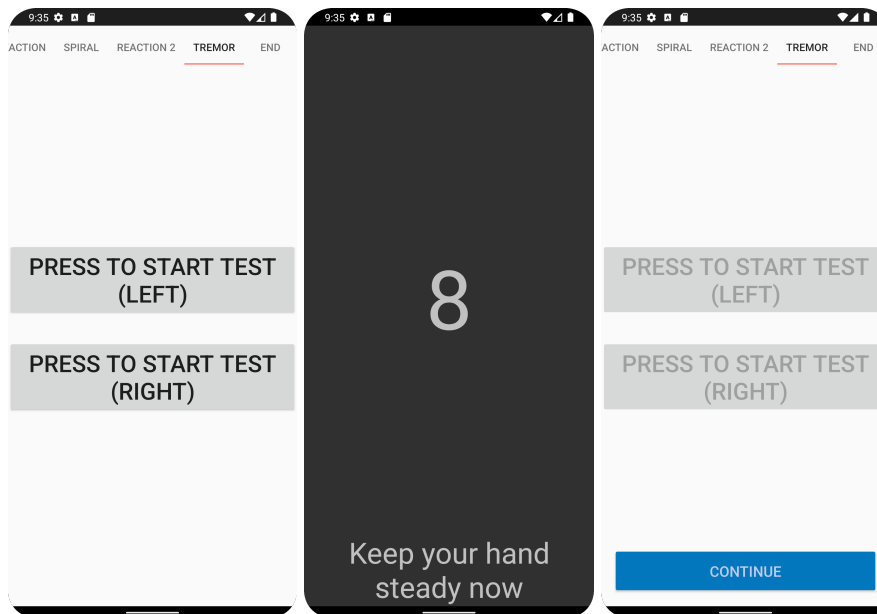


Figure 12. Screen view of the Tremor test.

During the test, the first point will be used for calibration - all movement changes will be accounted for from the initial position. Every 15-17 milliseconds a pulse goes through, saving acceleration on X, Y, and Z planes. The test will record overall time past and accelerations for all planes at a time.

## 4.5 Upload

When all tests have been done, the test data gets selected, processed, and sent to a separate database. As all test data at that point is saved in the application's local database, a processing step has to be taken to collect test data from each step. A separate thread starts, mapping the data to a format acceptable for the REST API. Once the data has been mapped,

a POST query is done to the REST API. After an HTTP204 message has been responded to, the Upload view will show a green checkmark, indicating a successful transmission. In case of an error, a red cross is displayed, indicating a failure in sending the data.

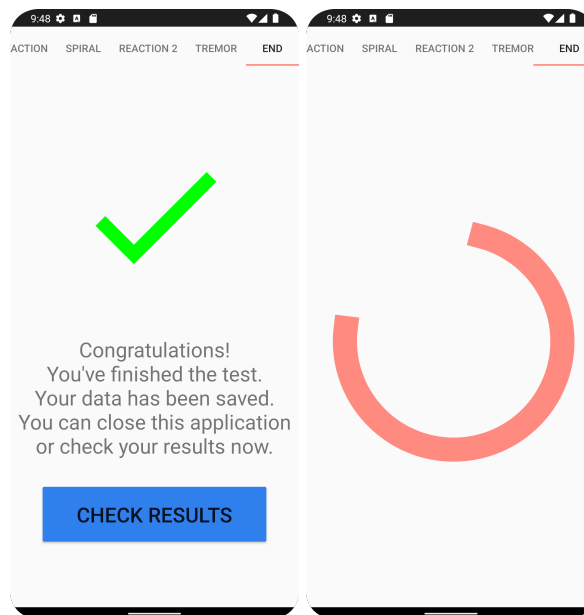


Figure 13. Screen view of upload progress and success view.

In the case of a successful transmission, a link will be provided with the device ID, allowing the user to navigate to the data dashboard to see their collected data. As there is no case for the test to stay open, it will also automatically close after 20 seconds. On re-entry to the application, a new instance would be started.

## 4.6 Tutorial

Accessibility has been one of the main focuses of the test. With design considerations made for the reaction time test and spiral drawing test in relation to the format of the devices, an important step in any testing process is the teaching of the process. Since minimal training of the user is preferred, every step and process had to be able to be understood by the subject while using the application.

In the test suite, this is resolved by tutorial insertion. Almost every new view loaded within the application is provided with a tutorial. With the first step of the application, a general overview is also provided. Every test has up to 5 steps of the tutorial, describing the positioning and steps of the test with a separate view to describe what data will be collected.



Figure 14. The user interface of the tutorials for each test.

## 5. REST API

Where the mobile application's functionality ends, the REST APIs begin. While the device holds on to all the test data until the application's reset or re-install, collecting data from it in a way that would not involve a processing layer would have deterred the design considerations selected when developing the mobile application. A central database for hosting all completed test data was devised.

The database technology selected was PostgreSQL. A free and open-source SQL solution that supports traditional relational database patterns and saves data as JSON blobs, allowing NoSQL approaches where applicable. As the test data was already saved as JSON and mapping it over to a SQL-friendly form factor, a decision was taken to use the NoSQL approach of saving the preformatted test data collections as JSON to a JSONB. This made for a simple database structure without loss in performance. As the spiral drawing test can produce up to 40 megabytes of data in the form of JSON, the processing speed of documents on both read and write had to be considered.

Modern database data pre-and post-processing for entry is typically handled by an application that can accept data from a foreign host and send it out to them. This application was written using a RESTful approach, where rules of communication between applications are written keeping to its patterns. This means that all read queries are solved as GET queries, all write queries are handled as POST queries and removals as DELETE. As the application does not need modification, no PUT or PATCH endpoints were created. The application was written in Kotlin, using Spring, Spring Boot. Data management was done using Spring Data JPA. Search functionality with Spring Specification API. The application's architecture has already been described further in the architecture section.

The application's main purpose is to relay data between the writing and reading parts of the solution. The writer is the mobile application, which is the only source of data, and the data dashboard is the sole reader. The API itself does no further processing of data other than a mapping process to separate the message sent and the data saved. Knowing this, the application consists of the following:

- TestHolderController - a controller with GET, POST, and DELETE endpoints for

saving the main test data. a TestHolder is described as the collection of all tests and is the message received from the mobile application and viewed by the data dashboard.

- TestHolderBasicController - a controller with a GET endpoint only. Used to describe a simplified view of the TestHolder object for more performant table rendering. A TestHolderBasic is described as a TestHolder without the JSON blob attached, decreasing the size of a message.
- TermsController - a controller with GET, POST, and DELETE endpoints for viewing and saving the mobile application terms of service agreement state. A Terms object is described as an object containing the boolean state of terms acceptance and the device id of the subject.

## 6. Data Dashboard

Sometimes described as a luxury, having an environment to manually check data integrity and verify test results became a very powerful tool even during the development of the test. Often, simple errors are overlooked, and data can seem right even though its underlying basis could be false. A simple tool to visually verify data and see a history of device tests creates accessibility for the test observer. A tool for visualizing test results with no code needed and a simple table to carry that data, the dashboard could be used in the future to observe test results to either remind test takers to retake the test or for manually verifying falsely performed tests.

As all test data is available publically and no personalization is expected, the application also had no authorization scheme planned for it. At the time of research, no functionalities exist for an administrative type that would necessitate an authorization layer or rights management.

The data dashboard comes with four views:

- A view containing all tests - a paginated table with the size of 20 tests for each page and a search function. Dashboard user can select to enter the details view of the selected test.
- A view containing the details of a selected test - every test is presented as a member of a tabulated view.
- A view containing all participated devices - a paginated table with the size of 20 devices per page for. Dashbooard user can enter the details view of a device.
- A view containing all device tests - a variant of the details view with a separate selector attached for browsing between tests.

### 6.1 All Tests View

The initial view the user is greeted with is the view that contains all tests. A view consisting of 20 tests is produced, ordered by time created by default. Each row of the view contains information of the device id, the UUID of the test, a timestamp for created time, and a link to the details view. An additional search box exists with the ability to search by

device ID in it. The search is realized as an HTML query to provide accurate and whole database-encompassing results.

Device ID	UUID	Created	Details
11c9d311-6144-4a8b-8e90-6f9dcd2bcee8	17469390-28f0-428b-85ae-47f9ff420ad9	18. detsember 2022, kell 16:17:19 GMT +1	<a href="#">DETAILS</a>
5f762c74-192c-48ec-bebb-8f68de923bef	c1df96e2-6db6-4b26-bf66-5d15feecae25	18. detsember 2022, kell 14:48:33 GMT +1	<a href="#">DETAILS</a>
32c7c8dd-59f6-4967-815f-703b1b6f1740	9f5d7468-9d5d-4d94-bb41-22c236514277	17. detsember 2022, kell 21:48:17 GMT +1	<a href="#">DETAILS</a>
32c7c8dd-59f6-4967-815f-703b1b6f1740	4e006d2d-922d-4cc5-bdb7-1627f41d35c6	16. detsember 2022, kell 13:38:07 GMT +1	<a href="#">DETAILS</a>
9fa272e1-4e03-4a6a-ade3-4af0c2b6d16a	26955d98-31a8-4e26-994a-967344aabb5d	14. detsember 2022, kell 13:21:42 GMT +1	<a href="#">DETAILS</a>
7d4624f-9926-43d5-bcb3-573149305f3a	1d17c86e-e315-4fcd-90b2-1812f0b67b9	6. detsember 2022, kell 23:50:31 GMT +1	<a href="#">DETAILS</a>

Figure 15. Browser capture of the initial view and main device table view.

Tables do not translate to an easily readable form factor in a mobile setting. As a function of the dashboard is to provide test subjects access to their test data, modifications were made depending on the screen's width to allow for a more readable build of the table. At a resolution below 1024 pixels in width, the table becomes transposed in view, allowing more space for reading data, albeit with more pages to scroll.

Device ID	11c9d311-6144-4a8b-8e90-6f9dcd2bcee8
UUID	17469390-28f0-428b-85ae-47f9ff420ad9
Created at	18. detsember 2022, kell 16:17:19 GMT +1
<a href="#">DETAILS</a>	
Device ID	5f762c74-192c-48ec-bebb-8f68de923bef
UUID	c1df96e2-6db6-4b26-bf66-5d15feecae25
Created at	18. detsember 2022, kell 14:48:33 GMT +1
<a href="#">DETAILS</a>	

Figure 16. Mobile device view of content presented in Figure 15.



## 6.2 Test Details View

For each test performed, a set of test data is collected. After navigating to the specific test, a call is made to the REST API to collect the details of that test. The metadata consisting of the questionnaire results, subject age, weight, gender, and main hand is displayed at the top of the view. The results of each test is displayed in a tabulated view with tabs following the test type. By default, the initial aiming test data is shown. A visualization is provided for each interaction, showing the reaction time for the interaction. When the user holds their cursor over the data point, a description of that interaction is shown. For every piece of the data visualized, the JSON it is collected from is also included for the test observer to quickly use for either explanation, debugging or input for their own work.

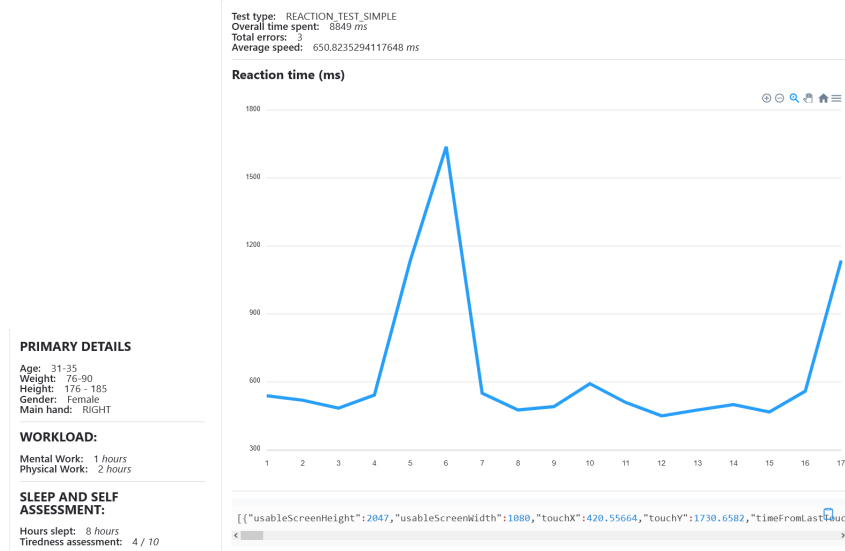


Figure 17. Browser capture of the test metadata and simple reaction test visualization.

When navigating to the Spiral draw test view, the user's painted spiral is rendered with data of points painted over the templates added.

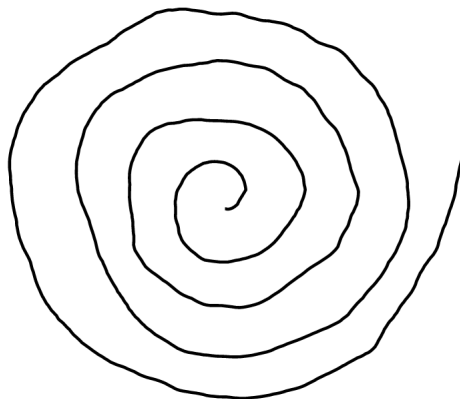


Figure 18. Browser capture of the ASD test painted by a test taker and rendered in the dashboard.

The second reaction test has the same visualizations as the initial reaction test, with a second visualization added for the collection of time since the expected color target's paint. Tremor test data is displayed in different tabs for each hand and holds visualizations of acceleration difference from the initial calibration point. A line graph is provided for each axis of movement together with the overall acceleration data. Average acceleration is recorded and provided separately.

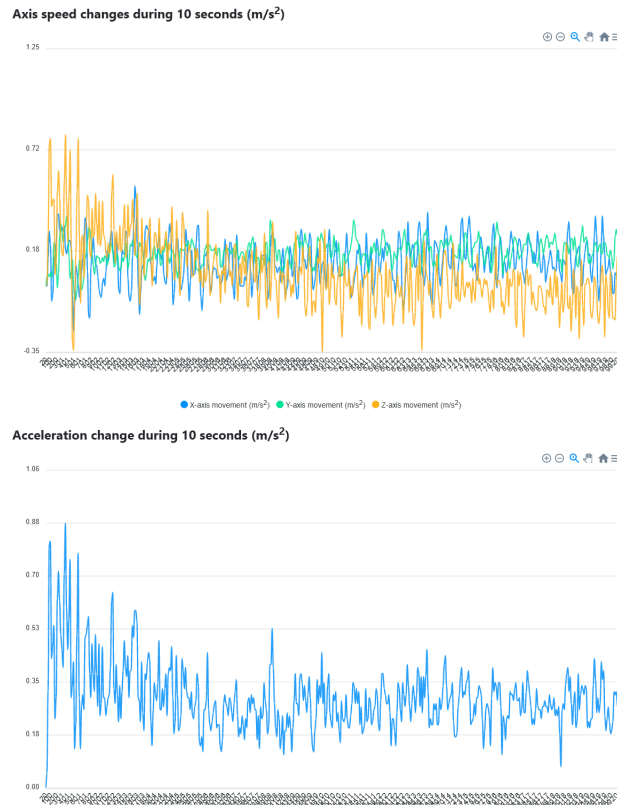


Figure 19. Browser capture of tremor test visualizations. Above shows velocity for x,y,z axis. Below shows absolute acceleration.

### 6.3 All Devices View

One of the main objectives of the dashboard is to allow the test observer to quickly identify devices that have partaken in the test. As the test takers and observers may not be in the same physical location, this can give the observer a method of validating tests as the results produce more expected results and data for analysis when the test taker has attempted the test at least twice during a day with optimal results after seven days of daily tests. The devices table allows ease of navigation to specific devices' tests for manual validation, provides a method of attempting contact with the group tested if possible, and gathers immediate visual validation of well-executed tests. This becomes important when removing faulty tests or debugging data pipelines built off the dataset.

The devices view holds three columns - device id, time of latest performed test, and a navigation button to the device details view. Using the same logic as the table for all tests, a transposition is done when the viewport width is smaller than 1024 pixels.

Device ID	Latest created at	Details
5f762c74-192c-48ec-bebb-8f68de923bef	18. detsember 2022, kell 14:48:33 GMT +1	<a href="#">DETAILS</a>
7fc435b0-daad-4aee-b7bc-a9b409b650b7	5. detsember 2022, kell 09:32:07 GMT +1	<a href="#">DETAILS</a>
a59cb38a-f377-4620-9870-4894bb86644e	29. november 2022, kell 21:17:56 GMT +1	<a href="#">DETAILS</a>
77e20530-8ebc-47e3-ae29-04c63900f623	29. november 2022, kell 12:22:15 GMT +1	<a href="#">DETAILS</a>
f1133b23-d7b1-47e1-9b35-2b67878a417f	28. november 2022, kell 11:01:25 GMT +1	<a href="#">DETAILS</a>
87fb3212-f455-4ffa-8146-3946e822bab6	27. november 2022, kell 18:35:24 GMT +1	<a href="#">DETAILS</a>

Figure 20. Browser capture of the device-based overview list view.

## 6.4 Device Tests Details View

When the user has navigated to a specific devices' details view, a result similar to the test details is provided. With a separate selector for test shown by the time taken, the user can navigate between tests without exiting the page.

The screenshot shows a web interface for device test details. At the top, there is a dropdown menu labeled "Select a test" with the value "3.12.2022..." and a close button. Below this, the "PRIMARY DETAILS" section lists: Age: 18-25, Weight: 61 - 75, Height: 186 - 190, Gender: Male, and Main hand: RIGHT. The "WORKLOAD:" section shows Mental Work: 0 hours and Physical Work: 0 hours. The "SLEEP AND SELF ASSESSMENT:" section shows Hours slept: 12 hours and Tiredness assessment: 4 / 10. A JSON object is displayed below: {"height": "186 - 190", "weight": "61 - 75", "age": "18-25", "gender": "Male", "tiredScale": 4, "sleepScale": 12, "mentalWor...}. At the bottom, there is a tabbed navigation bar with five tabs: "Reaction test (simple)" (selected), "Spiral draw test", "Reaction Test (Advanced)", "Tremor test (Left Hand)", and "Tremor test (Right Hand)".

Figure 21. Browser capture of the test metadata and tab view of the device-based test details view.

## 7. Assessing Statistical Value of Data

While the application, database, REST API, and data dashboard compose the data collection solution, a reusable script was assembled for data extraction as well. Using Python, NumPy, and Pandas, the script selects all tests from the database if none have been created for the day earlier, saves, filters, and processes them, leaving a JSON file of each collection of tests with a corresponding Pickle file for reusing the data.

As the goal of the work was to present a data collection scheme and a dataset for use in future research, the immediate expectations set during this work were not in large data processing but in proving the usability of that data. Due to this, an expectation was set that by using statistical hypotheses following common logic, and data could be proven for use in future research. Kinematic features could be proved usable in research if these hypotheses' can be proved.

### 7.1 Experimental Setting

Before any data collection could be performed, terms of use were defined with the Tallinn University Board of Ethics 12.05.2021 decision nr. 12 for use during test collection alongside a tablet-based data collection process. The content of the terms of use will be shared in Appendix 2. The document outlines assurances of only non-personal data being collected and highlights types and methods of metadata collection, test suite progress, and data storage logic. Before any data collection took place in schools, an agreement was found with the parties participating. An example of the document sent to the management of schools can be found in Appendix 3.

In the classroom setting, the tablet and mobile application-based solutions were presented to the students simultaneously. In classrooms of 17-25 students between the ages of 15-19, in a 45-minute session, up to 5 tests were collected using the tablet approach. Up to 30 tests could be collected during the 45-minute session. Overall, 157 tests were collected in two school visits.

One of the expectations of the tests was that not all results would be perfect or conform to the standards set by the mobile application. For these, a filtering process was made. The

process looks for spiral draw tests with less than 500 data points recorded, specific device IDs for debugging device removal, and tremor tests with more than 3 seconds per square meter movement. After filtering, a symmetry check is performed. Symmetry is defined by allowing only test sets with all tests included to be ones available as the dataset. After filtering and checking for symmetry, 132 tests eligible for use were found.

The data setup and feature extraction were done in Python. While a frontend exists for the database, a decision was taken to make a direct connection to the database for faster data download and processing. As the first step, the data download saves a record in JSON format of all the datasets. These will be transformed into dataframes and saved as Pickle files. After filtering and symmetry ensuring, feature extraction can be performed.

While the reaction tests or the tremor test did not acquire any new computed features, the ASD tests' amount of features increased from 12 variables to 69.

In all, five datasets are formed every time the pipeline is run. With reports presented immediately of the current data and accompanying JSONs and pickles, the datasets are immediately available for use outside of the pipeline or as a next step in the process.

### 7.1.1 Feature Engineering (ASD test)

Raw time series described in the previous section: finger position (x- and y-coordinates) and timestamps can be used to compute an infinite number of features. The feature set was computed based on the work of Valla [9], where novel tremor-related features were introduced.

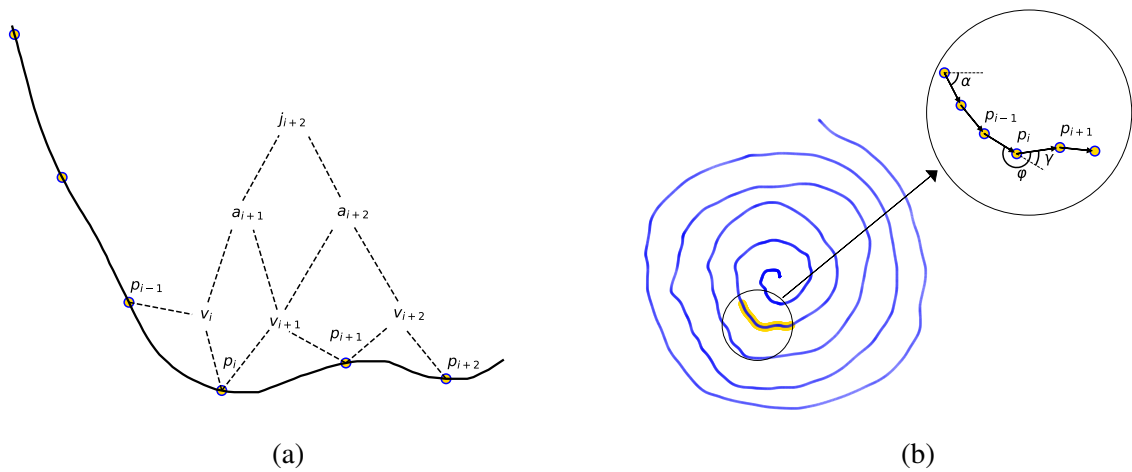


Figure 22. Visual representation of the differential-type (a) and angular-type (b) features [9].

Given a respective timestamp, we can calculate the velocity of the position vector  $\vec{r} =$

$[p_i, p_{i+1}]$ . In other words, velocity is the rate at which displacement of the position vector changes with respect to time. Similarly, acceleration was computed as the rate of change in velocity and jerk as the rate of change in acceleration with respect to time. Following the sequence, we considered up to the sixth time derivative of the position vector. There are no universally accepted names for higher time derivatives of the displacement than the fourth. The terms snap, crackle and pop have been used in the literature previously to describe fourth, fifth, and sixth-time derivatives of displacement[31]. The angular feature set was also enriched with up to a third of respective time derivatives.

Studies conducted by [32][9] proved a connection between higher levels of derivatives possessing high discriminating power to distinguish patients from control subjects. In it were also demonstrated features called motion mass parameters that could allow for machine learning techniques to detect mental fatigue. Having proved its use in the original research and model-building efforts done later[9][8], these parameters became of use. Nõmm introduced motion mass parameters in 2013 to describe the amount and smoothness of motion of a limb or some other group of joints[33]. For each kinematic, geometric, and pressure parameter that changes during the test sum of the absolute values at each observation point would be computed, allowing for additional statistics.

## 7.2 Hypothesis Control

To prove the usability of the dataset, four hypotheses were set:

- A test performed in the morning will get a different reaction speed and tremor than one performed in the evening.
- A test performed with self-assessed higher fatigue will result in a different reaction speed than one with lower.
- A test performed by a person with more mental work done will result in a different reaction speed and tremor.
- A test performed by a person with more physical work done will result in a different reaction speed and tremor.

As these hypotheses' are set, the final goal of them is to not prove any substantive results from tests. Meaning the hypotheses should be able to prove through a Welch t-test and Mann-Whitney U-test that there is significance to the data difference between the two groups. These were chosen as existing work has proven the connection to some hypotheses. While a selection of tests exist for simple reaction time test comparison to time awake, Cain proved through a Stroop test that the performance of subjects degraded significantly when compared to time spent awake [34]. A presumption exists with the first hypothesis of

the subject not sleeping during the daytime. As time awake does not exist as a data point, this was taken as an expectation. Two groups were prepared: people who had given the test in the time period between 05:00 AM and 12:00 PM as morning tests and all who had given the test in a time after as evening.

A person's self-perceived fatigue state was one of the questions asked when initiating the test. While research does not follow the scale set in this test, the previous examination of the hypothesis has been done, showing a relation between perceived fatigue relation to reaction tests. Aldughini found in Multiple Sclerosis patients that perceived fatigue is associated with performance fatigability assessed using an attentional task [35]. As a reaction test is an attentional test, Aldughini's method of assessment was also backed by multiple other tests for MS patients and a scale that was not completely in the realm of self-assessment. Fifty-two participants (mean  $\pm$  SD age,  $46.8 \pm 10.1$  years) completed their study where three measures of performance fatigability were used: percent change in meters walked from first to last minute of the 6-Minute Walk Test, percent change in the force exerted from first to last trial on a repetitive maximal hand grip test, and response speed variability on the Continuous Performance Test. Perceived physical and cognitive fatigue was measured using the NFI-MS, a 10-item summary scale[35]. As the mobile application does not expect an overseeing party, the value of the reaction speed and tremor became of interest. Two groups were formed, where a perceived fatigue rate of below and including three was taken as not fatigued and a fatigue rate of 7 and above as fatigued.

Mental fatigue is one of the key points of interest in the tests. As proof of data significance, mental work assessment became one of the hypotheses of interest while proving the dataset. Senkiv proved a difference in spiral drawing test performance in mentally fatigued people using machine learning techniques focusing on acceleration mass as they found it to be one with a higher Fisher score [8]. Migliaccio has found a difference in reaction time in sportsmen under different levels of mental fatigue using hand-eye tasks [13]. The marker of cognitive tasks performed by the number of hours was used for grouping. Two groups were formed, where the non-fatigued were separated into a group as people who reported 0 hours of mental work performed before taking the test and fatigued as the group who had performed at least 1 hour of cognitive tasks during the day before taking the test.

While Senkiv did not find a difference in physical fatigue affecting the model performance of their spiral drawing test results, anecdotal perceived results of tremor tests and reaction tests could suggest it has an effect. As the difficulty of the reported performance was not asked, physical work done provided a question of whether any physical work done during the day could affect the performance of tests in independent samples. Using the same scale



Label	Welch T-Test P-value	Mann-Whitney U-Test P-value
day_night_reaction	<b>5.050047e-03</b>	<b>2.241194e-10</b>
day_night_reaction_advanced	3.729564e-01	4.350033e-01
day_night_tremor_left	<b>9.862045e-102</b>	<b>7.984776e-94</b>
day_night_tremor_right	0.000000e+00	0.000000e+00
day_night_spiral_alpha_mass	9.440125e-02	1.681855e-01
day_night_spiral_phi_angle_mass	1.182206e-01	1.934897e-01
day_night_spiral_yaw_mass	9.451562e-02	1.660144e-01
day_night_spiral_crackle_mass	7.593188e-01	1.983778e-01

Table 1. Table of day and night Welch and Mann Whitney U-Test p-values. Values under alpha are presented in **bold**.

as for mental fatigue, groups were formed.

Independent, non-symmetrical groups formed on all hypotheses, on which Welch t-test and Mann-Whitney U-test were performed. Extracting the p-value as proof of significance. A significance below the alpha of 0.05 was accepted as a significant difference. Datasets from both reaction test types were used with a focus on reaction time from render being the base. For the advanced test, time from correctly colored render was ignored. Tremor test datasets also came into use. Using the same groups, absolute acceleration was used for analysis. For all hypothesis tests, alpha acceleration mass was used to form a Welch t-test result and Mann-Whitney u-test result. As Valla found novel features with significance in the angular properties of a spiral drawing test and higher derivative acceleration results [9], a selection of these was also used for p-value analysis. Phi and yaw mass was used for angular data, and crackle as a higher derivative of acceleration was also selected. Mass is the sum of data points, which Valla used in their research as a representation of smoothness. A significance recorded by a p-value below alpha of 0.05 would be accepted as proof of significance.

Welch t-test and Mann-Whitney U-tests were performed using the scikit-learn library for Python [36].

### 7.3 Hypothesis Control Results

Analysis of the first hypothesis saw significance in the reaction test suggesting a difference between simple aiming reactions between day and night. While this was not represented in the advanced reaction test results, the dataset does suggest a difference in the test with a more intuitive approach. These results display that a subject's simple reaction time may vary between morning and night but not so much when the element of the following color is introduced. While the left-hand tremor test suggests much significance in variability, it

Label	Welch T-Test P-value	Mann-Whitney U-Test P-value
self_assessed_reaction	0.126990	<b>0.012958</b>
self_assessed_reaction_advanced	0.417227	0.452280
self_assessed_tremor_left	0.000000	0.000000
self_assessed_tremor_right	0.000000	0.000000
self_assessed_spiral_alpha_mass	<b>0.000171</b>	<b>0.000406</b>
self_assessed_spiral_phi_angle_mass	<b>0.000166</b>	<b>0.000368</b>
self_assessed_spiral_yaw_mass	<b>0.000171</b>	<b>0.000406</b>
self_assessed_spiral_crackle_mass	<b>0.003122</b>	<b>0.007699</b>

Table 2. Table of self-assessed fatigue grouped Welch and Mann Whitney U-Test p-values. Values under alpha are presented in **bold**.

Label	Welch T-Test P-value	Mann-Whitney U-Test P-value
mental_work_reaction	4.329119e-01	8.175350e-01
mental_work_reaction_advanced	<b>8.088434e-03</b>	<b>5.993804e-03</b>
mental_work_tremor_left	<b>4.745755e-193</b>	<b>1.438157e-188</b>
mental_work_tremor_right	<b>3.766779e-294</b>	<b>1.681287e-286</b>
mental_work_spiral_alpha_mass	8.045922e-01	9.385510e-01
mental_work_spiral_phi_angle_mass	7.674213e-01	9.357636e-01
mental_work_spiral_yaw_mass	8.040879e-01	9.497071e-01
mental_work_spiral_crackle_mass	1.496283e-01	<b>4.972513e-02</b>

Table 3. Table of mental work grouped Welch and Mann Whitney U-Test p-values. Values under value are presented in **bold**.

seems that the number of data points available with the calculation methods used could no longer provide a small enough value for the right hand, defaulting the value to 0. The ASD test, however, produced no significance in examined data points in relation to the alpha set. As such, it can be determined that in the scope of this dataset, a simple aiming reaction test and tremor could be used to prove the hypothesis, but the ASD test and advanced reaction test results do not support the hypothesis.

The self-assessed fatigue hypothesis produced a result where neither the reaction tests could produce significance in fatigue. This could suggest that the method selected for fatigue self-assessment could vary too much in scope. While already limited in grouping to assign tighter groups, there either is not enough data in relation to reaction tests, or these values do not support the hypothesis set. Tremor tests produced p-values close enough to zero that the used method of calculation failed to produce a small enough number, opting for 0. While reaction tests did not prove the hypothesis, the spiral drawing test results definitely did. With p-values well below alpha, significance can be proven between the two groups already with the amount and quality of data collected.

While self-assessed fatigue produced results in reaction tests that could not produce significance, the grouping of mental work associated with already mild mental work

Label	Welch T-Test P-value	Mann-Whitney U-Test P-value
physical_work_reaction	5.970050e-01	8.642523e-01
physical_work_reaction_advanced	6.774431e-01	4.995745e-01
physical_work_tremor_left	<b>4.183457e-34</b>	<b>3.771181e-55</b>
physical_work_tremor_right	<b>3.358324e-243</b>	<b>2.994988e-246</b>
physical_work_spiral_alpha_mass	<b>1.034633e-02</b>	<b>8.019125e-03</b>
physical_work_spiral_phi_angle_mass	<b>7.836895e-03</b>	<b>5.429116e-03</b>
physical_work_spiral_yaw_mass	<b>1.035886e-02</b>	<b>8.019125e-03</b>
physical_work_spiral_crackle_mass	1.838123e-01	7.134533e-01

Table 4. Table of physical work grouped Welch and Mann Whitney U-Test p-values. Values under value are presented in **bold**.

before taking the test to significant in advanced reaction test results, this suggests a relation between higher attention-requiring task performance degradation after a cognitive workload. Significance was found in tremor test results. While angular properties did not produce significance, one was found in derivative kinematic features. While barely below alpha, the significance could be found in the third derivative of acceleration, crackle. This could suggest that while angular properties of drawing the spiral in a mentally fatigued state do not differ, the speed at which the test is performed could.

Whereas mental work seemed to affect properties related to acceleration in drawing the spiral draw test, the used grouping of physical work was found to affect angular properties to a significant level. While the results for ASD produced significant results, reaction time tests could not. As cognitive workloads also differ between the groups asked to test, physical workloads could differ greatly. While the test did not set requirements for the time after physical work nor provided a set of control for it, this could already explain the lack of significance. Tremor tests again produced results that could be used for further research.

Overall, none of the hypothesis set could be proven on the grounds of all test results suggesting significance, but each test was found to produce results in some tests.

## 7.4 Linear Association Assessment

To find a correlation between groups and data results, correlation tests were run for all datasets using Pearson and Spearman correlation. As a linear scale, the expectations were to find correlations between groups and data points recorded. A correlation map was taken for all tests with further focus on the value extracted spiral drawing tests where the various kinematic feature masses were extracted. For reaction and tremor tests, all available data was inserted to a correlation map though filters describing the physical properties of the

	timeFromLastTouch	tired_group	sleep_binary	physical_binary	mental_binary
timeFromLastTouch	1.000000	-0.021442	0.075141	0.006153	-0.003705
tired_group	-0.021442	1.000000	-0.043701	-0.012975	0.000340
sleep_binary	0.075141	-0.043701	1.000000	-0.071157	-0.043857
physical_binary	0.006153	-0.012975	-0.071157	1.000000	0.280468
mental_binary	-0.003705	0.000340	-0.043857	0.280468	1.000000

Table 5. Table of Spearman correlations between simple reaction test times and numeric groupings observed by metadata.

	timeFromLastTouch	tired_group	sleep_binary	physical_binary	mental_binary
timeFromLastTouch	1.000000	-0.039321	0.143416	-0.052727	0.022821
tired_group	-0.039321	1.000000	0.031026	0.088884	0.094914
sleep_binary	0.143416	0.031026	1.000000	0.030107	-0.048223
physical_binary	-0.052727	0.088884	0.030107	1.000000	0.252353
mental_binary	0.022821	0.094914	-0.048223	0.252353	1.000000

Table 6. Table of Spearman correlations between advanced reaction test times and numeric groupings observed by metadata.

screen were implemented. A correlation of 0.3 upward was deemed as a sufficient level of correlation.

The simple aiming reaction test did not produce data points with acceptable correlation measures. While mental workload and physical workload reached a correlation of 0.28, no significant markers of correlations were found between metadata-based groupings and reaction time where timeFromLastTouch describes reaction time.

While a correlation of 0.15 was found between reaction time and time slept, the low correlations in the realm of 0.09 did not produce results that would suggest using a machine learning approach for the advanced aiming reaction test.

Where abs describe the absolute acceleration of the device, a statistically significant correlation of 0.28 for the left hand and 0.25 was found between reported time slept and a negative correlation of -0.23 for the left and -0.19 right hand for self-assessed fatigue. While approaching the imposed 0.3 limits for use as features in a machine learning model, the correlation was not high enough to warrant further approach. While the right-hand tremor did not produce a high correlation between data points, the tremor test performed on the right hand produced a correlation of 0.22.

The ASD test produced results with a correlation above 0.3. With a negative correlation of 0.41, alpha phi and yaw mass provided results of statistical significance. Results were

	abs	tired_group	sleep_binary	physical_binary	mental_binary
abs	1.000000	-0.198563	0.281660	-0.057951	0.084226
tired_group	-0.198563	1.000000	-0.040737	-0.026743	0.010390
sleep_binary	0.281660	-0.040737	1.000000	-0.065447	-0.045035
physical_binary	-0.057951	-0.026743	-0.065447	1.000000	0.279102
mental_binary	0.084226	0.010390	-0.045035	0.279102	1.000000

Table 7. Table of Spearman correlations between tremor tests for right-hand absolute acceleration and numeric groupings observed by metadata.

	abs	tired_group	sleep_binary	physical_binary	mental_binary
abs	1.000000	-0.182681	0.257731	-0.099094	0.223907
tired_group	-0.182681	1.000000	-0.040985	-0.026067	0.009925
sleep_binary	0.257731	-0.040985	1.000000	-0.064459	-0.044319
physical_binary	-0.099094	-0.026067	-0.064459	1.000000	0.278271
mental_binary	0.223907	0.009925	-0.044319	0.278271	1.000000

Table 8. Table of Spearman correlations between tremor tests for absolute left-hand acceleration and numeric groupings observed by metadata.

also produced from the higher derivative acceleration features. While acceleration mass was found at -0.33, crackle and snap produced results of -0.35 and -0.37. The collection of these classifiers suggested a correlation that would warrant further research.

## 7.5 Machine Learning

To classify fatigue, the self-assessed fatigue state was used for grouping. While similar to the second tested hypothesis, the groupings were modified to include more tests. The groupings are as follows:

- Self-perceived fatigue below and including a score of 3 was chosen as not tired.
- Self-perceived fatigue above and including 6 was chosen as tired.

In all, 84 tests were selected for training and validation. Of that, 47 were tests deemed as tired and 37 as not tired.

Six machine learning classifiers were used for the classification and research:

- Logistic Regression (LR)
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Decision Tree (DT)
- Random Forest (RF)
- AdaBoost (AB)

These were trained and cross-validated in a nested [37] k-fold loop [9]. Training and validation of the classifiers were performed using the scikit-learn library for Python [36]. Accuracy, precision, sensitivity, specificity, and f1-score were used to assess the performance of classifiers. Specificity (true negative rate) refers to the ability of the test to identify subjects without the disease correctly; sensitivity (re-call or true positive rate) — refers to the ability of the test to identify those patients with the disease correctly. As used before by Valla in a similar context and owing to wide use in medical diagnostic settings, the choice was made based on previous research [9]. Although correlation results

could serve as an initial feature selection method, the wrapper method, namely the SVM recursive feature elimination (SVM-RFE), was applied as a feature selection step for machine learning based classification [38].

In the following Table 9, averaged over all the iterations of a 5-fold cross-validation scheme the mean accuracy, precision, sensitivity, specificity, and f1-score values are reported.

Table 9. Fatigue classification cross-validated model performance. The best scores for each test are presented in **bold**.

Test	Features	Classifier	$P_{acc}$	$P_{prec}$	$P_{sen}$	$P_{spec}$	$P_{f1}$
ASD test	$\phi\_angle\_mass$ , $crackle\_mass$	LR	65.29%	65.29%	87.55%	37.50%	<b>73.96%</b>
		RF	60.59%	63.00%	76.44%	40.00%	68.10%
		KNN	<b>66.69%</b>	<b>73.30%</b>	72.00%	<b>58.57%</b>	70.03%
		SVM	62.94%	62.06%	<b>89.56%</b>	29.29%	65.46%
		DT	55.81%	60.74%	76.67%	29.64%	65.46%
		AB	59.41%	62.26%	74.44%	40.36%	67.01%
RTS test	wasHitOnTarget, timeFromLastTouch	LR	<b>59.34%</b>	60.16%	<b>78.44%</b>	34.64%	<b>68.01%</b>
		RF	45.37%	49.34%	54.89%	31.79%	51.07%
		KNN	55.88%	<b>60.24%</b>	61.11%	<b>48.57%</b>	60.20%
		SVM	52.35%	56.88%	63.11%	37.86%	58.96%
		DT	47.65%	51.02%	48.67%	46.07%	49.24%
		AB	54.63%	58.00%	63.11%	42.50%	59.83%
RTA test	timeFromLastTouch, timeFromFirstCorrectColorRender	LR	<b>66.76%</b>	65.23%	<b>86.87%</b>	40.36%	<b>74.40%</b>
		RF	58.38%	62.44%	65.56%	47.86%	63.58%
		KNN	61.91%	<b>65.50%</b>	70.22%	<b>51.07%</b>	67.22%
		SVM	63.09%	62.63%	86.67%	31.79%	72.08%
		DT	49.92%	55.41%	50.67%	48.21%	52.61%
		AB	54.93%	58.42%	63.33%	42.86%	60.35%
Tremor test	$y$ , $abs$	LR	58.30%	58.26%	<b>89.33%</b>	18.93%	<b>70.34%</b>
		RF	<b>68.31%</b>	64.40%	58.89%	<b>57.14%</b>	59.74%
		KNN	55.88%	58.82%	67.56%	40.36%	62.27%
		SVM	51.18%	53.68%	89.11%	0.025%	66.97%
		DT	56.03%	65.71%	63.11%	46.43%	60.46%
		AB	65.51%	<b>69.59%</b>	71.55%	56.79%	69.36%

Based on the cross-validated performance the best-performing classifiers with the respective feature set were trained on the whole dataset (split 1/3 test/train set) for further analysis. The highest results were achieved on the ASD and RTA tests. Table 10 describes the final models with a respective performance review.

With the highest sensitivity, the RTA test showed promising results classifying tired sessions from not-tired ones. The [timeFromFirstCorrectColorRender"] feature achieved the highest discriminating power. Measured in both RTS and RTA tests ["wasHitOnTarget"] feature performed poorly (accuracy scores 47-51%). It proved to be less critical in the current experimental setting. Overall, the RTS test was significantly less successful in the fatigue classification task.

Table 10. Best performing machine learning models for fatigue classification.

Test	Features	Classifier	Performance (%)				Confusion Matrix											
			$P_{acc}$	$P_{sen}$	$P_{spec}$	$P_{f1}$												
ASD	$\phi\_angle\_mass$ , $crackle\_mass$	KNN $k=12$	66.69	83.33	46.67	73.17	<table border="1"> <tr> <td rowspan="2">Actual</td> <td>not_tired</td> <td>7</td> <td>8</td> </tr> <tr> <td>tired</td> <td>3</td> <td>15</td> </tr> <tr> <td colspan="2">Predicted</td> <td>not_tired</td> <td>tired</td> </tr> </table>	Actual	not_tired	7	8	tired	3	15	Predicted		not_tired	tired
Actual	not_tired	7	8															
	tired	3	15															
Predicted		not_tired	tired															
RTS	timeFromLast-Touch	RF $n = 12$	57.58	66.67	46.67	63.16	<table border="1"> <tr> <td rowspan="2">Actual</td> <td>not_tired</td> <td>7</td> <td>8</td> </tr> <tr> <td>tired</td> <td>6</td> <td>12</td> </tr> <tr> <td colspan="2">Predicted</td> <td>not_tired</td> <td>tired</td> </tr> </table>	Actual	not_tired	7	8	tired	6	12	Predicted		not_tired	tired
Actual	not_tired	7	8															
	tired	6	12															
Predicted		not_tired	tired															
RTA	timeFrom-LastTouch, timeFromFirst-CorrectColor-Render	LR $C=0.001$ , penalty=12	66.76	86.87	40.36	74.40	<table border="1"> <tr> <td rowspan="2">Actual</td> <td>not_tired</td> <td>6</td> <td>9</td> </tr> <tr> <td>tired</td> <td>2</td> <td>16</td> </tr> <tr> <td colspan="2">Predicted</td> <td>not_tired</td> <td>tired</td> </tr> </table>	Actual	not_tired	6	9	tired	2	16	Predicted		not_tired	tired
Actual	not_tired	6	9															
	tired	2	16															
Predicted		not_tired	tired															
Tremor	$y, abs$	LR $C = 0.001$ , penalty=12	69.70	66.67	73.33	70.59	<table border="1"> <tr> <td rowspan="2">Actual</td> <td>not_tired</td> <td>11</td> <td>4</td> </tr> <tr> <td>tired</td> <td>6</td> <td>12</td> </tr> <tr> <td colspan="2">Predicted</td> <td>not_tired</td> <td>tired</td> </tr> </table>	Actual	not_tired	11	4	tired	6	12	Predicted		not_tired	tired
Actual	not_tired	11	4															
	tired	6	12															
Predicted		not_tired	tired															

The best-performing model for the ASD test was kNN with 12 nearest neighbors. Angular and high-order derivative features introduced in [9] were among the best-performing predictors in ASD-based fatigue classification. Using a similar methodology to classify Parkinson’s disease, Valla found an accuracy of up to 73.7% [9] for the Parkinson’s Disease Handwriting (PaHaW) database[39], and accuracy of 84.3% for the Drawing and Handwriting Parkinson’s Disease dataset (DraWritePD) [10].

The highest accuracy was achieved with the Tremor test. The tremor values were defined as an asymmetry between the left and the right-hand points, computed as the subtraction.

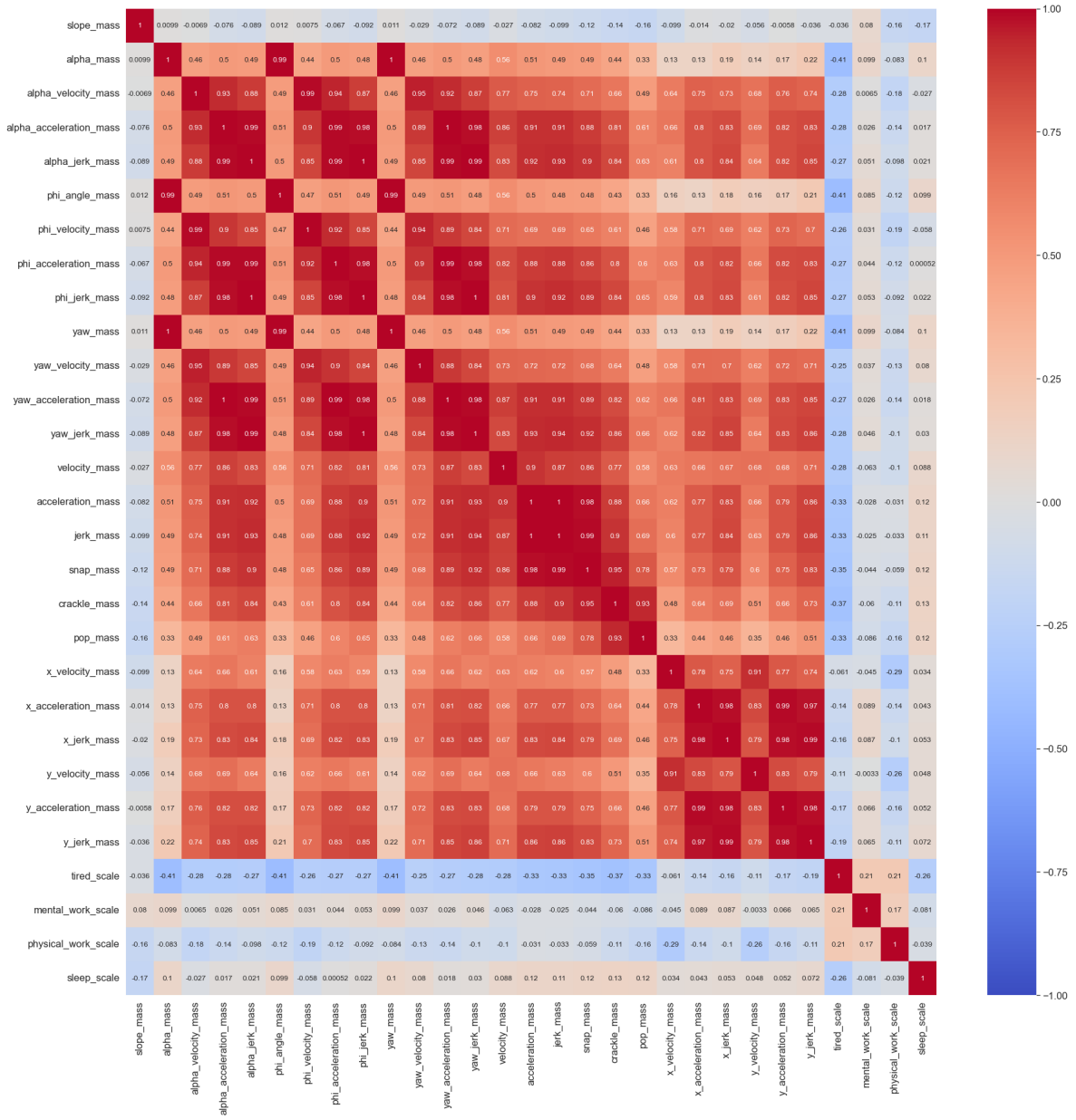


Figure 23. Image of Spearman correlations between ASD tests classifiers and numeric groupings observed by metadata.



## 8. Discussion

The application and dataset have provided much discussion for all participating in its development. One of the application's main bonuses is the breadth of future research - the dataset created has proven accuracy and significance for four hypotheses. With much more anecdotally devised and at least ten attempted during development, the dataset has much to offer for future research even at the current point.

With more oversight, the author believes that more accurate models could be built within a month's work, if not less, using existing, available code. Thus building the basis for future thesis' and other academic work, the ease of use for data collection has, in the authors' and their supervisors' eyes, a novel avenue for data collection.

Many future academic works were already devised in the datasets' review stages. From producing more accurate results, creating an iOS application to modifying approaches and hypotheses set towards the dataset were devised as plans for future work. The researchers expect their work on the software suite not to be the last within the realm of fatigue assessment using the produced mobile application and suite in TalTech.

Though the machine learning model produced acceptable accuracy, the researchers believe that results could be made further viable through a more accurate oversight of test takers. The ability to perform a fatigue test at any place at any time produces a double-edged sword. While the ease of access to the test provides an easier path to collecting results and performing the test, it also discredits the standard limitations of a scientific test. Most control methods that exist during a classic scientific scenario were not available. Yet, even with the current setting, the machine learning models carry merit. This means an expectation of even higher accuracy could be reached with more control added to the experimental setting.

While pressure data was mentioned in the feature extraction for ASD tests, results showed a significant difference in how Android devices would record pressure. Some devices lacked advanced touch pressure sensing features, opting for the operating system to record pressure as a boolean value. As most tests were found with pressure values of either 0 or 1, no further assessment of pressure-related features would be done.

During the tremor test machine learning model building, models with high levels of sensitivity were found. With levels reaching 90%, further research should be done to study the applicability of tremor tests for fatigue classification. Studying asymmetry of hand tremors was proven a subject for further research not only in current problem domain. As an example of other motorics related domains, studying stroke victims using a similar setup could produce valuable results.

## 9. Summary

A framework within the realm of fatigue assessment was proposed to solve an issue of data collection and analysis. With an Android smartphone application for test takers, a REST API for result processing, and a data dashboard for manual review, a solution was devised for an easier data collection scheme in the view of spiral drawing, aiming, and tremor tests.

Three applications were created, with a fourth assembled for analysis. Kotlin, Typescript, and Python were used to assemble a software suite for fatigue recognition research. Tests for reaction time through aiming tests, tremor, and spiral drawing tests were devised. Datasets were assembled in a dynamic manner that could be used for further research. With the usability of the suite in mind, the capability for further research and expansion and the dataset were devised.

Our research concluded that the time resources of data acquisition were verified to be greater than the techniques used currently. The mobile application could be downloaded, studied, and completed in 5 minutes and ran again later, whereas the tablet-based alternative required training and oversight at each test. A 45-minute session, with two tablets available, produced up to five new results, and the mobile application produced 30 new test results. Overall, 132 usable tests were collected for research.

Five separate datasets were built. The usability of the datasets was proven through the use of a simple hypothesis set upon it with tests proving the significance of data within the dataset. To prove the capability of the spiral drawing test (ASD) within the data, a machine learning model was built with 66.69% accuracy, 73.30% precision, 72.00% sensitivity, 58.57% specificity, and 70.03% F1-score for determining fatigued and non-fatigued participants in a test set without a strict control group.

Machine learning models were implemented for other tests as well. Tremor test results provided results with even higher accuracy than the ASD test at 68.31% and the RTS the lowest at 59.37% from the battery of models implemented. The ASD, RTA, and tremor tests provided sensitivity ratings between 86 to 89 percent.

Thus, the created software suite provided a collection of usable datasets for future research. The solution could potentially serve as a fatigue management tool with further development.

## Bibliography

- [1] Roger M. Enoka and Jacques Duchateau. “Muscle fatigue: What, why and how it influences muscle function”. In: *Journal of Physiology* 586.1 (2008), pp. 11–23. ISSN: 00223751. DOI: 10.1113/jphysiol.2007.139477.
- [2] Maarten A.S. Boksem, Theo F. Meijman, and Monique M. Lorist. “Mental fatigue, motivation and action monitoring”. In: *Biological Psychology* 72.2 (2006), pp. 123–132. ISSN: 03010511. DOI: 10.1016/j.biopsycho.2005.08.007.
- [3] Maarten A.S. Boksem, Theo F. Meijman, and Monique M. Lorist. “Effects of mental fatigue on attention: An ERP study”. In: *Cognitive Brain Research* 25.1 (2005), pp. 107–116. ISSN: 09266410. DOI: 10.1016/j.cogbrainres.2005.04.011.
- [4] Ying-ying Zhang et al. “Determinants of compassion satisfaction, compassion fatigue and burn out in nursing: A correlative meta-analysis”. In: *Medicine* 97.26 (2018), pp. 1–7.
- [5] Xinyun Hu and Gabriel Lodewijks. “Exploration of the effects of task-related fatigue on eye-motion features and its value in improving driver fatigue-related technology”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 80 (2021), pp. 150–171. ISSN: 13698478. DOI: 10.1016/j.trf.2021.03.014. URL: <https://doi.org/10.1016/j.trf.2021.03.014>.
- [6] M.T. Tarata et al. “P13-9 SEMG derived parameters vs blood oxygen saturation in monitoring neuromuscular fatigue”. In: *Clinical Neurophysiology* 121 (2010), S180. ISSN: 13882457. DOI: 10.1016/s1388-2457(10)60740-7. URL: [http://dx.doi.org/10.1016/s1388-2457\(10\)60740-7](http://dx.doi.org/10.1016/s1388-2457(10)60740-7).
- [7] Seth L Pullman. “Spiral analysis: a new technique for measuring tremor with a digitizing tablet”. In: *Movement disorders* 13.S3 (1998), pp. 85–89.
- [8] Olesja Senkiv, Sven Nõmm, and Aaro Toomela. “Applicability of Spiral Drawing Test for Mental Fatigue Modelling”. In: *IFAC-PapersOnLine* 51.34 (2019), pp. 190–195. ISSN: 24058963. DOI: 10.1016/j.ifacol.2019.01.064.
- [9] Elli Valla et al. “Tremor-related feature engineering for machine learning based Parkinson’s disease diagnostics”. In: *Biomedical Signal Processing and Control* 75 (2022), p. 103551.

- [10] Sven Nömm et al. “Deep CNN Based classification of the archimedes spiral drawing tests to support diagnostics of the Parkinson’s disease”. In: *IFAC-PapersOnLine* 53.5 (2020), pp. 260–264.
- [11] Erik Dzotsenidze et al. “Generative Adversarial Networks as a Data Augmentation Tool for CNN-Based Parkinson’s Disease Diagnostics”. In: *IFAC-PapersOnLine* 55.29 (2022), pp. 108–113.
- [12] Ana Cristina de Bem Alves et al. “Role of adenosine A2A receptors in the central fatigue of neurodegenerative diseases”. In: *Journal of Caffeine and Adenosine Research* 9.4 (2019), pp. 145–156.
- [13] Gian Mario Migliaccio et al. “Effects of Mental Fatigue on Reaction Time in Sportsmen”. In: *International Journal of Environmental Research and Public Health* 19.21 (2022), p. 14360. ISSN: 16604601. DOI: 10.3390/ijerph192114360.
- [14] Diogo Coutinho et al. “Exploring the effects of mental and muscular fatigue in soccer players’ performance”. In: *Human Movement Science* 58.March (2018), pp. 287–296. ISSN: 18727646. DOI: 10.1016/j.humov.2018.03.004. URL: <https://doi.org/10.1016/j.humov.2018.03.004>.
- [15] JetBrains Kotlin Foundation. *Kotlin*. <https://kotlinlang.org/assets/kotlin-media-kit.pdf>. Nov. 2022.
- [16] Matias Martinez and Bruno Gois Mateus. “Why did developers migrate Android Applications from Java to Kotlin”. In: *IEEE Transactions on Software Engineering* (2021). ISSN: 19393520. DOI: 10.1109/TSE.2021.3120367. arXiv: 2003.12730.
- [17] Microsoft. *Typescript*. <https://www.typescriptlang.org/>. Nov. 2022.
- [18] JetBrains Kotlin Foundation. *Kotlin Multiplatform*. <https://kotlinlang.org/docs/multiplatform.html>. Nov. 2022.
- [19] VMware. *Spring (framework)*. <https://spring.io/projects/spring-framework/>. Nov. 2022.
- [20] Evan You. *VueJS*. <https://vuejs.org/>. Dec. 2022.
- [21] Open Collective. *SvelteKit*. <https://kit.svelte.dev/docs/introduction>. Nov. 2022.
- [22] Gian Mario Migliaccio et al. “Effects of Mental Fatigue on Reaction Time in Sportsmen”. In: *International Journal of Environmental Research and Public Health* 19.21 (2022), p. 14360. ISSN: 16604601. DOI: 10.3390/ijerph192114360.
- [23] OCJ Lippold. “The tremor in fatigue”. In: *Human muscle fatigue: Physiological mechanisms* (1981), pp. 234–248.

- [24] Francesco Budini et al. “Tremor, finger and hand dexterity and force steadiness, do not change after mental fatigue in healthy humans”. In: *Plos one* 17.8 (2022), e0272033.
- [25] Lauren B. Krupp et al. “The fatigue severity scale: Application to patients with multiple sclerosis and systemic lupus erythematosus”. In: *Archives of Neurology* 46.10 (1989), pp. 1121–1123. ISSN: 15383687. DOI: 10 . 1001 / archneur . 1989 . 00520460115022.
- [26] Liliane Lins-Kusterer et al. “Validity and reliability of the 36-Item Short Form Health Survey questionnaire version 2 among people living with HIV in Brazil”. In: *Brazilian Journal of Infectious Diseases* 23.5 (2019), pp. 313–321. ISSN: 16784391. DOI: 10 . 1016 / j . bjid . 2019 . 08 . 001. URL: <https://doi.org/10.1016/j.bjid.2019.08.001>.
- [27] John E Ware Jr and Cathy Donald Sherbourne. “The MOS 36-item short-form health survey (SF-36): I. Conceptual framework and item selection”. In: *Medical care* (1992), pp. 473–483.
- [28] M. Worm-Smeitink et al. “The assessment of fatigue: Psychometric qualities and norms for the Checklist individual strength”. In: *Journal of Psychosomatic Research* 98.May (2017), pp. 40–46. ISSN: 18791360. DOI: 10 . 1016 / j . jpsychores . 2017 . 05 . 007. URL: <http://dx.doi.org/10.1016/j.jpsychores.2017.05.007>.
- [29] J Ridley Stroop. “Studies of interference in serial verbal reactions.” In: *Journal of experimental psychology* 18.6 (1935), p. 643.
- [30] Olesja Senkiv. “Fatigue recognition modelling”. In: (2018).
- [31] Reza N Jazar. *Advanced dynamics: Rigid body, multibody, and aerospace applications*. John Wiley & Sons, 2011.
- [32] Sergei Zarembo et al. “CNN Based Analysis of the Luria’s Alternating Series Test for Parkinson’s Disease Diagnostics”. In: *Asian Conference on Intelligent Information and Database Systems*. Springer. 2021, pp. 3–13.
- [33] Sven Nomm and Aaro Toomela. “An alternative approach to measure quantity and smoothness of the human limb motions.” In: *Estonian Journal of Engineering* 19.4 (2013).
- [34] Sean W Cain et al. “One night of sleep deprivation affects reaction time, but not interference or facilitation in a Stroop task”. In: *Brain and cognition* 76.1 (2011), pp. 37–42.

- [35] Mayis Aldughmi, Jared Bruce, and Catherine F Siengsukon. “Relationship between fatigability and perceived fatigue measured using the neurological fatigue index in people with multiple sclerosis”. In: *International journal of MS care* 19.5 (2017), pp. 232–239.
- [36] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [37] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning. 2nd Edition*. Springer Series in Statistics. Springer, 2002.
- [38] Isabelle Guyon et al. “Gene selection for cancer classification using support vector machines”. In: *Machine learning* 46.1 (2002), pp. 389–422.
- [39] Peter Drotár et al. “Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson’s disease”. In: *Artificial intelligence in Medicine* 67 (2016), pp. 39–46.



# Appendix 1 - Non-exclusive licence for reproduction and publication of a graduation thesis <sup>1</sup>

I Ain-Joonas Toose

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "A novel data collection solution through digital fine motor skill assessment for fatigue visualization and analytics", supervised by Elli Valla and Sven Nõmm
  - 1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
  - 1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

09.01.2023

---

<sup>1</sup>The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

**Appendix 2 - Terms of Service PDF provided to  
test taker before being allowed to start tests**

# RESEARCH PARTICIPATION INFORMATION SHEET

Welcome to the Fatigue Test Application Terms of Use agreement. For purposes of this agreement, “App” refers to our mobile application in which users are asked to complete the questionnaire and three fine-motor skill related tests. The terms “we,” “us,” and “our” refer to the Fatigue Test App. “You” refers to you, as a participant in this research.

The following Terms of Use apply when you use the App on your mobile device.

Please review the following terms carefully and signify your agreement to these Terms of Use at the bottom by clicking Agree. **If you do not agree to be bound by these Terms of Use in their entirety, you may not access or use the App.**

## I. INTRODUCTION

This research is conducted by researchers at the Tallinn University of Technology Department of Software Science. The main scope of the study is to develop a framework for human motor function and cognitive impairment analysis. Movement and neurological disorders pose a significant burden on the healthcare system. Our goal is to provide

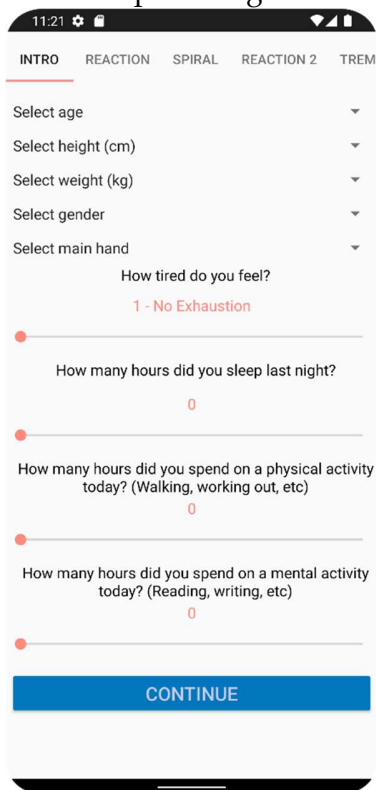


Figure 1: Questionnaire

decision support tools to help clinicians with data collection, diagnostics, and treatment processes. The more data we collect, the more accurate and reliable applications we can develop. We are thankful for any contribution. Participation is entirely voluntary, and you can withdraw your data anytime.

## II. INFORMATION WE COLLECT

We collect “Non-Personal Information”. **Non-Personal Information** includes information that cannot be used to personally identify you, such as anonymous usage data, and general demographic information we may collect. More specifically, App consists of four parts:

1) **Questionnaire**, with data to be collected:

- a. age (interval)
- b. height (interval),
- c. weight (interval),
- d. gender,
- e. the current perceived state of fatigue (scale 0-12),
- f. the number of hours spent on a physical activity,
- g. the number of hours spent on a mental activity,
- h. dominant hand.

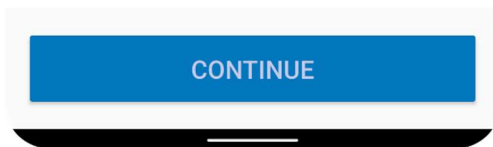
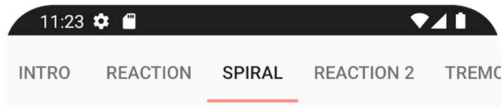


Figure 3: Spiral drawing test

2) **Reaction test (level 1).** The user presses the circle on the screen of their mobile device with their finger.

Signals to be collected:

- a. reaction time,
- b. test duration,
- c. error rate (distance between circle and pressure point).

3) **Spiral drawing test.**

The user is presented with a spiral template that the user must follow when drawing with their finger.

Signals to be recorded:

- a. kinematic and dynamic parameters:
  - i. screen coordinates,
  - ii. time,
  - ii. pressure

applied on the screen.

- b. error rate (difference between template and user drawing with bitmap accuracy),
- c. test duration.

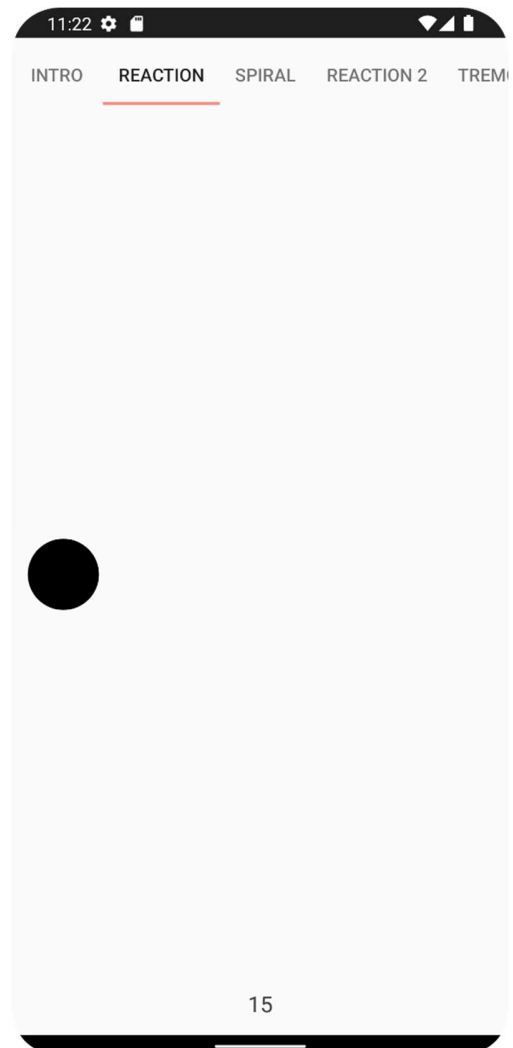


Figure 2: Reaction test (level 1)

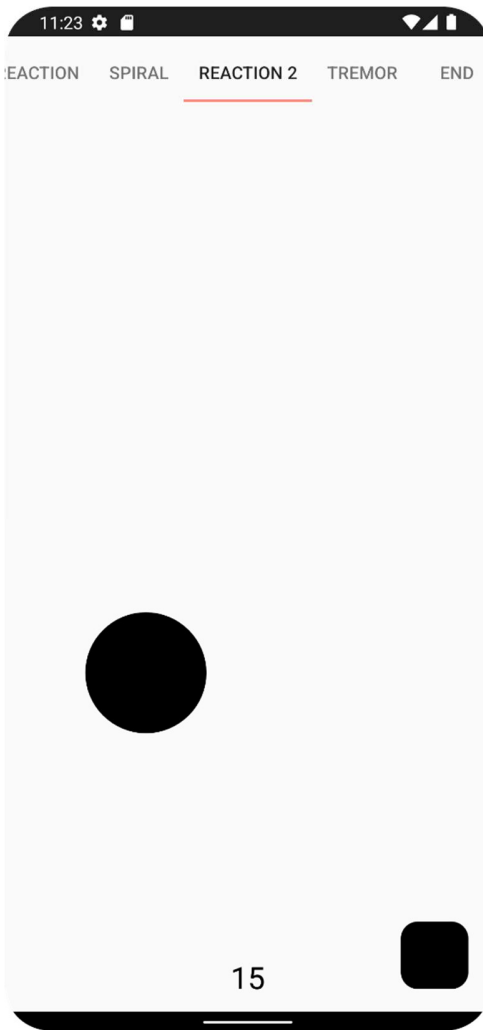


Figure 4: Reaction test (level 2)

4) **Reaction test (level 2)**. In the second reaction test the color codes add an additional complexity. In this case, the user only must press the circle marked with a color code. Signals to be recorded:

- reaction time,
- test duration,
- error rate (distance between circle and pressure point).

5) **Tremor test**. The user is asked to hold the device in front of him with the arm stretched out for 10 seconds.

- axial derivations recorded by the accelerometer,
- test duration.

To activate the App, you do not need to submit any Personal Information.

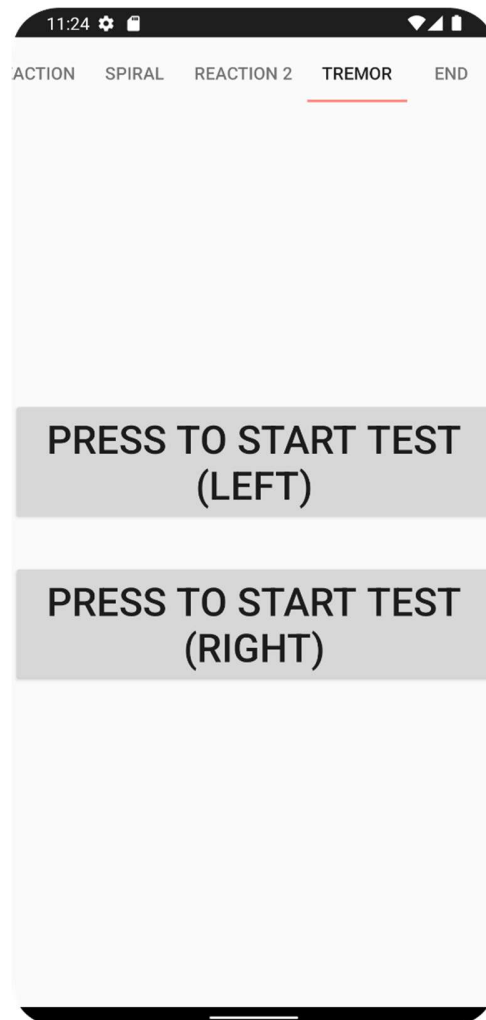


Figure 5: Tremor test

### III. HOW WE USE AND SHARE INFORMATION

The collected data will be used as research data by the TalTech University the Department of Software Science to further the knowledge around cognitive impairment and human motor function analysis.

### IV. HOW WE STORE AND PROTECT INFORMATION

We further protect your information from potential security breaches by implementing encrypted data transfer over a secure socket layer connection and storing it in a secured database. The data will become accessible over an off-site application programming interface by authorized users. The data storage and management flow diagram are depicted in Figure 6. However, these measures do not guarantee that your information will not be accessed, disclosed, altered, or destroyed by a breach of such firewalls and secure server software. By using our App, you acknowledge that you understand and agree to assume these risks.

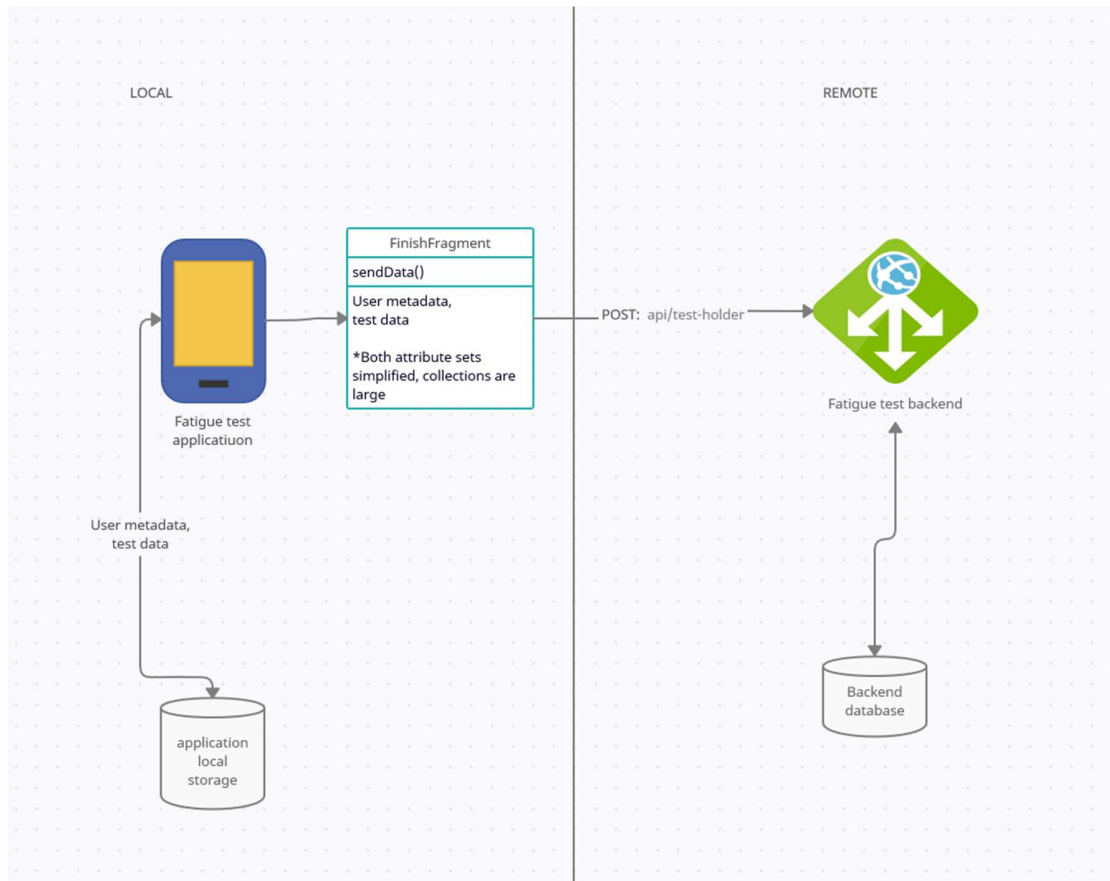


Figure 6: Data storage and management.

We keep information for as long as we need it for our research. We decide how long we need information on a case-by-case basis.

#### V. YOUR RIGHTS REGARDING THE USE OF YOUR DATA

You have the right to erasure. It means that you can ask to delete your data from our databases at any time.

#### VI. CONTACT US

If you have any technical questions and concerns regarding the practices of this App, please contact us by sending an email to [elli.valla@taltech.ee](mailto:elli.valla@taltech.ee).

Last Updated: This Information Sheet was last updated on 22.08.2022.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS RESEARCH PARTICIPATION INFORMATION SHEET , UNDERSTAND THE TERMS OF USE, AND WILL BE BOUND BY THESE TERMS AND CONDITIONS. YOU FURTHER ACKNOWLEDGE THAT THESE TERMS OF USE REPRESENT THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US AND THAT IT SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

**Appendix 3 - PDF of research request sent to secondary education schools in Tallinn.**



## *Uuring: Kuidas ennetada vaimset taandarengut 2021-2026*

### **Lugupeetud koolijuht!**

**Kutsume Teie kooli osalema Tallinna Ülikooli uurimisprojekti *Kultuuri-, bioloogiliste ja arenguliste tegurite roll kognitiivse reservi mehhanismides ja kognitiivse taandarengu ennetamises*** (uuringu vastutav täitja, Tallinna Ülikooli professor Aaro Toomela).

Inimkond vananeb ja vananemisega kaasneb vaimse võimekuse vähenemine kuni selle kadumiseni veel inimese eluajal. Käesoleva uuringu kõige olulisemaks eesmärgiks on vananemise ja ajukahjustusega seotud vaimse taandarengu mehhanismide mõistmine ning selle alusel taandarengu aeglustamist toetavate tegevuste parem planeerimine. Kõik vaimse võimekuse olulised aspektid omandavad inimesed elu jooksul õppides. **Selleks, et mõista vaimset taandarengut peab kindlasti aru saama ka sellest, kuidas mõistus on arenenud.** Laste uurimine on väga vajalik selleks, et hilisema võimaliku taandarengu mehhanisme mõista. Ühelt poolt võimaldab see avastada viise, kuidas taandarengut ennetada või pidurdada ja teiselt poolt viise, kuidas erinevatel põhjustel toimunud taandarengu tagajärjel kaotatud vaimseid funktsioone neuropsühholoogilise taastusraviga jälle tagasi saada.

Teadusuuringutest on ka teada, et täiskasvanutel algab vaimne taandareng erinevalt sõltuvalt saavutatud kõrgeimast tasemest. Osadel inimestel algab taandareng palju varem ja toimub palju kiiremini, kui teistel. Selliste erinevuste mõistmiseks on tingimata vaja paremini tunda vaimse arengu üldisi mehhanisme lastel. *Vastavalt keskendub meie kompleksse uuringu üks suund laste vaimse arengu mehhanismide paremale tundmaõppimisele.* Seetõttu on meie uuringust kasu ka lastele. **Uuringus avastame uusi võimalusi, kuidas vaimse arengu probleeme varem ära tunda ning millisel viisil võimalikke arenguprobleeme ennetada.**

### **Eetilised küsimused**

Uuringu eetiline külg on meie jaoks väga oluline. *Uuringu läbiviijad garanteerivad isikuandmete puutumatuset. Kõik kogutavad andmed on anonüümsed.* Uuringu läbiviimiseks on luba Tallinna Ülikooli eetikakomiteelt (12. mai 2021 otsus nr 12).

### **Mida ja kuidas soovime Teie koolis teha**

Meie väga mitmekülgse uuringu üks allteema on inimese peenmotoorsete oskuste arengu parem mõistmine. Teie koolis soovime algklasside ja lõpuklasside õpilastel hinnata peenmotoorseid oskusi. Need testid (erinevad joonistamise ülesanded) esitatakse individuaalselt tahvelarvutil. Aega ühele õpilasele kulub orienteeruvalt 15 minutit. Lisaks esitame rea küsimusi kirjalikuks vastamiseks. Need küsimused võimaldavad hinnata õpilase üldist vaimse arengu taset. Kirjalikud vastused kogutakse grupis. Orienteeruv vastamise aeg samuti umbes 15 minutit.

*Küsimuste korral võite pöörduda projekti juhi Aaro Toomela poole (kirjutada aadressil [aaro.toomela@ilu.ee](mailto:aaro.toomela@ilu.ee)).*

Lugupidamisega,  
Aaro Toomela, TLÜ kultuuri- ja neuropsühholoogia professor  
projekti juht