NTNU
Norwegian University of
Science and Technology

An Application for Detection of Dyslexia

Tore Angell Petersen

Master of Science in Computer Science
Submission date: April 2018
Supervisor: John Krogstie, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

Dyslexia is a disability that has accumulated more and more attention over the past years. Being able to detect dyslexia early on is essential for decreasing the negative effects it has on people diagnosed with this disability. This master thesis deals with creating a database system of an existing application meant to be used as a screening tool in the process of detecting dyslexia. This is done by developing a web-service, a database and a website to extend the application. The result of this process is an application with a fully functional data collection system and a way to show this data graphically. This master thesis provides the application with a data collection layer that further helps scientists to collect information to study dyslexia. Hopefully, when this application is deployed, the average age of dyslexia detection will be significantly reduced and it will support the study of dyslexia.

The research questions defined for this project are twofold. First, they put focus on the collection of personal data. Secondly, they address the possibilities of implementing a web-service to collect information from the application without any disturbance to the usability in the application Magno. These questions are answered in this report through requirement analysis, design and creation to extend the Magno application and evaluation of the extended system.

Sammendrag

Dysleksi er en funksjonshemning som har fått mer og mer oppmerksomhet de siste årene. Å være i stand til å oppdage dysleksi tidlig er viktig for å redusere de negative effektene det har på personer som er diagnostisert med denne funksjonshemmingen. Denne masteroppgaven omhandler å skape et databasesystem til en eksisterende applikasjon ment å bli brukt som et verktøy i prosessen med å oppdage dysleksi. Dette gjøres ved å utvikle en webtjeneste, en database og et nettsted for å utvide applikasjonen. Resultatet av denne prosessen er en applikasjon med et fullt funksjonelt datainnsamlingssystem og en måte å vise disse dataene grafisk. Denne masteroppgaven gir applikasjonen et datasamlingslag som videre hjelper forskere til å samle inn informasjon for å studere dysleksi. Forhåpentligvis, når denne applikasjonen blir distribuert, vil gjennomsnittsalderen for dysleksi-deteksjon bli betydelig redusert, og den vil støtte studiet av dysleksi.

Forskningsspørsmålene utarbeidet i dette prosjektet er todelt. For det først legger de vekt på innsamling av personopplysninger. For det andre adresserer de mulighetene for å implementere en webtjeneste for å samle inn informasjon fra applikasjonen uten forstyrrelser av brukbarheten i søknaden Magno. Disse spørsmålene vil besvares i denne oppgaven gjennom analyse av kravspesifikasjonen, design og utvikling for å utvide Magno-applikasjonen og evaluering av det utvidede systemet.

Acknowledgments

The work presented in this thesis is a part of my master's degree in computer science at the Norwegian University of Science and Technology (NTNU). The project is conducted under the Department of Computer Science at the Faculty of Information Technology and Electrical Engineering, under the supervision of Professor J. Krogstie. The project was done in collaboration with Professor H. Sigmundsson at the Department of Psychology at NTNU.

I would like to thank Professor John Krogstie for his support and advisory during the making of this master thesis. Without his help, this thesis would not have been possible. I would also like to thank Professor Hermundur Sigmundsson and Kaja Egset for contribute with their knowledge in the field of dyslexia and dyslexia detection. My girlfriend deserve to be thanked for being there for me during both good and bad times. Here care and motivation is what have allowed me to focus on the project during stressful situations.

A special thanks is directed towards my family for their unprecedented support and for providing me with the foundation needed in order to complete my master's degree here at NTNU.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Big data is a fast evolving area within computer science that describes any voluminous amount of structured, semi structured and unstructured data that has the potential to be mined for information. The data can come from several different sources, such at such as business sales records, real-time sensors that are used in the internet of things or the collected results of scientific experiments. The idea behind the application Magno is that it can be used for possible early detection of dyslexia and also collect vast amounts of data to study the correlation between dyslexia and other aspects of the human being.

Reading is complex. It requires our brain to connect letters to sound and put those sounds in the right order, and pull the words together into sentences so that we can read and understand what is written. People with dyslexia have trouble matching the letters they see on a page with the sounds those letters and combination of letters make. But these difficulties does not have any connection to their overall intelligence. In fact, dyslexia is an unexpected difficulty in reading in an individual that has the intelligence to be a much better reader[1]. The importance of detection dyslexia early on is undoubted, as proper intervention has up to 80 percent effectiveness when applied to children in the 1st and 2nd grade of primary school[2].

1.1 Motivation

With the importance of data collection and early dyslexia detection being as is, the combination of the two makes for an interesting and rewarding master's thesis. The opportunity to not only provide the public with a tool that helps scientist further research on the effects of enabling early dyslexia detection not based on the ability to read or write, but also develop and implement a data structure that contributes to the collection of information, is unparalleled. Being able to participate in the development of this program, could potentially contribute to the detection of several more cases of dyslexia, and hopefully increasing the quality of life of one or more individuals suffering from dyslexia.

My thesis focuses primarily on developing an application programming interface and a database that collect data and process data. The web-service is the link between data collection and the scientist that uses the data, and is important to ensure usability and accessibility of the data. My motivation for focusing on this aspect of the development is to ensure that this program can be used for the masses and also big scientific researches.

1.2 Project Objective

The main objective of this project is to provide a system meant to collect data from the Magno application without interference to the user interface. In a larger sense, one might say that my objective is to ensure steady collection of the information, in order to promote use of the program. By tailoring the data collection of a system to meet the users needs, one lowers the threshold for taking the program into use. The ultimate objective of this project is to help in early dyslexia detection, and providing the researchers with a tool to do so.

1.3 Project Description

This project is a continuation of the master thesis "App for Early Detection of Dyslexia" conducted in the spring of 2016 [3] and the master thesis

"Magno: An Application for Detection of Dyslexia" [2], as well as the authors' specialization project "Designing an Application for Detection of Dyslexia" conducted in the fall of 2016 [4]. "App for Early Detection of Dyslexia" was a master thesis concerning the development and implementation of a digital test that might help in dyslexia screening. "Designing an Application for Detection of Dyslexia" was a specialization project that had to do with creating a concept for a user interface to go along with the implemented screening test. "Magno: An Application for Detection of Dyslexia" was regarding the further development of the user interface design, and implementation of this user interface. They identified any problem areas in the existing user interface design, received from previous work, found solutions to these problem areas, and implementing a finished user interface design. This project covers further development of the application, and implementing a server to handle the data collection. This project is meant to identify what data to collect without any personal security issues. This without interfering the existing code, received from previous work. Also finding solutions to these problem areas, and implementing a finished system design.

1.4 Thesis Outline

This thesis consists of six different chapters. This chapter is the first chapter, which includes the introduction. Chapter 2 contains information regarding the research approach used in this project, and chapter 3 introduces all previous work done in relation to this project. In chapter 4 the implementation process and the finished graphical user interface that has been implemented are explained. Chapter 5 contains an evaluation of the work done in this project, and chapter 6 encompasses discussion around the project, a conclusion and any further work that might be needed. In addition, this thesis includes an appendix with one chapter.

Chapter 2

Research Approach

In this chapter, we will describe the chosen research approach. In section 2.1 the different research questions are presented, in section 2.2 we will discuss the research paradigm for this project and in section 2.3 we will discuss the different selected research models and methodologies. The final deliverables are declared in section 2.4

2.1 Research Question

This master thesis aims to develop a web-service for data collection and implementing the finished system into an existing application for screening for dyslexia [2]. The research questions to complement this master's thesis description are as follows:

RQ-1 What are the problem areas with developing the web-service in this project?

RQ-2 What are the problem areas with collecting personal data?

RQ-3 How can we compliment existing code with the implementation of the new web-service?

- **RQ-3.1** What technical tools are best suited to aid in implementing a web-service for data collection, based on previously implemented functionality?

2.2 Research Method

In this project I have chosen to use a combination of two research methods. These are literature study and design-science. The method were selected to obtain a solid foundation in the field through both empirical data and theory. This allows for better justification of the choices taken throughout the project. Due to the technological nature of the project, design-science is a fitting research method as I wish to answer my research question through development of a proof-of-concept solution. Design-Science is a set of techniques used in research on information-systems and -technology. Research which utilize these techniques follow a process where one acquire new knowledge through innovation, or close-to-realizable artifacts. This knowledge is used to analyze and reflect upon the effects of interaction with these artifacts [5]. The analysis is to be done after the literature study is complete, giving insights into previous experiments in this project's domain, as well as related theory to increase the quality of the first design iteration.

The research questions will be answered in this report by investigating the entire system through Design Science Research [5] and review of literature and related work. The research conducted in this project had several phases. Starting from the Knowledge Base constructed during the start of this project, some small changes was made to the application by tapping into the Knowledge Base as illustrated by the right side of Figure 2.1.

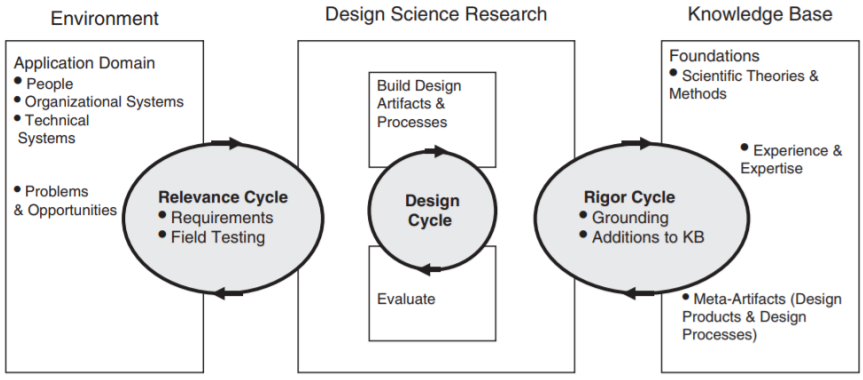


Figure 2.1: Design Science Research Cycles

The second, and most extensive phase, was the iterations of Design Sci-

ence Research illustrated by the left side of Figure 2.1. Here, the extended RestAPI, database and web site was implemented through several iterations of the Design Cycle. The requirements were implemented through Build Design Artifacts & Processes and was iteratively tested and evaluated by the developer. The Research Process adapted from Oates [6] with my personal process indicated in red, can be seen in Figure 2.2. This figure highlights the main strategies, the data generation methods and data analysis methodology applied during this research project.

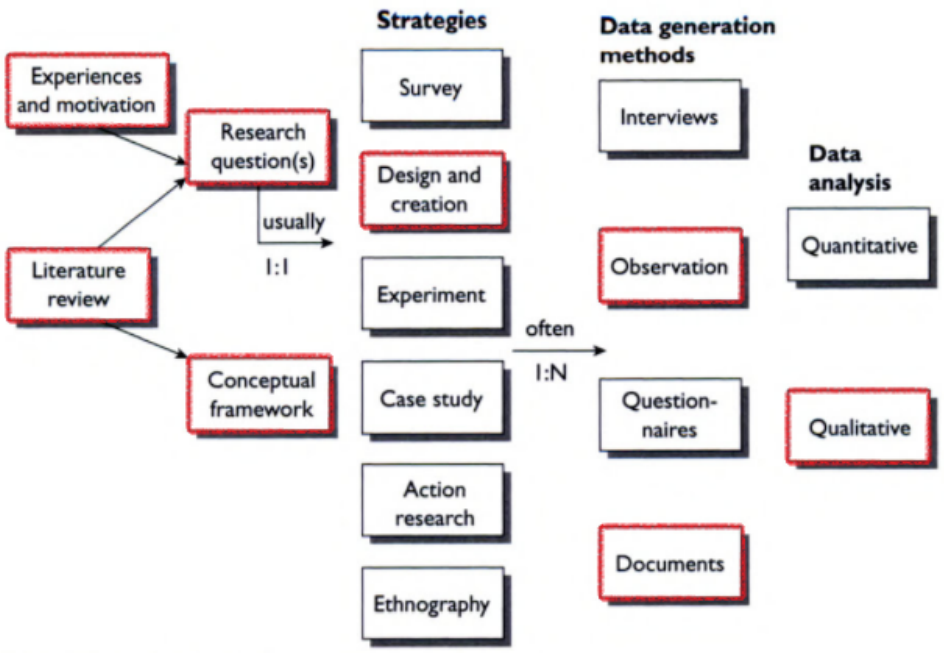


Figure 2.2: Model of The Research Process (adapted from [6])

2.2.1 Participants and Stakeholders

Participants of some form of research are the ones who have participated, contributed, or are otherwise featured in the research. Stakeholders are the ones who have something to either gain or lose depending on the outcome of the research.

The participants in this project are primarily the researchers, the authors of this master thesis, who contribute with their work and research on the subject at hand. The project supervisor, Professor J. Krogstie, a professor at NTNU, contributes with his expertise and experience in research in the field of information systems, and is also a participant. Other than the researchers and the project supervisor, external experts on the subjects of dyslexia and dyslexia detection have also been consulted. These external experts should also be viewed as participants of this project. The stakeholders of this project are the researchers, as the research will impact the grade given, and I am interested in presenting work that will reflect our work ethics and abilities. The project supervisor and NTNU will want this project to be conducted properly and with high quality, as the work reflects the quality of the university and its professors, and the publication will represent the quality and determination of the students who attend NTNU. Potential dyslectics, teachers, and doctors specializing in dyslexia and other reading disabilities will have something to gain as early detection of dyslexia minimizes potential problems both inside and outside of the classroom. Detecting dyslexia early will make life easier for the ones who are diagnosed, teachers who have to teach them, and the doctors who treat and diagnose them. Psychology students and researchers can use this application as a tool and an aid in research concerning dyslexia.

2.3 Final Deliverables and Dissemination

The project will conclude in the delivery of this master thesis. In addition a working version of the implemented application will be delivered, along with all coding for the different parts of the project. The master thesis will show an insight into the workings of the application, and how the application itself can be used as is, or further development in order to enable extended functionality.

Chapter 3

Previous Work And Requirements

Section 3.1 will describe what dyslexia is and it introduces the idea that some of the things causing dyslexia might lead to motor deficits as well. Also that the anatomical differences between the brains of known dyslexics and non dyslexics are presented. Section 3.2 contains an explanation of how dyslexia is detected in Norway today, and I give a thorough walk through of the applications that is the precursor to this project. The requirements are presented in the last section; section 3.3

3.1 Dyslexia

The disability causing reading abilities to be significantly worse than expected given the IQ, is called dyslexia. It is a neurological syndrome and has no connection with a person's intelligence other than poor orthographic[1] skills [7]. There exist several definitions of dyslexia as it manifests itself a little differently for each individual.

Some of the definitions are as follows:
«Dyslexia is a specific learning disability that is neurobiological in origin. It is characterised by difficulties with accurate and/or fluent word recognition and by poor spelling and decoding abilities. These difficulties typically result from a deficit in the phonological component of language that is often

[1]The study of spelling and how letters combine to represent sounds and form words.

unexpected in relation to other cognitive abilities and the provision of effective classroom instruction. Secondary consequences may include problems in reading comprehension and reduced reading experience that can impede growth of vocabulary and background knowledge.»
- *International Dyslexia Association* [8]

«Dyslexia is a specific learning difficulty that mainly affects the development of literacy and language related skills. It is likely to be present at birth and to be life-long in its effects. It is characterised by difficulties with phonological processing, rapid naming, working memory, processing speed, and the automatic development of skills that may not match up to an individual's other cognitive abilities. It tends to be resistant to conventional teaching methods, but its effect can be mitigated by appropriately specific intervention, including the application of information technology and supportive counselling.»
- *British Dyslexia Association* [9]

«A general term for disorders that involve difficulty in learning to read or interpret words, letters, and other symbols, but that do not affect general intelligence.»
- *Oxford dictionary* [10]

The definitions have similarities, but they are not equal. A similarity is that none of these definitions mention anything about what the cause of dyslexia is, only the symptoms. There are many different opinions on what causes dyslexia, but it is generally agreed upon that dyslexia is a genetic disorder, and therefore hereditary[11].

There have been conducted several studies that link dyslexia to other motor skills. In 2005, H. Sigmundsson published a research paper suggesting a link between dyslexia and poor performance in reaction time tests. The tests conducted showed that dyslexics on average responded .19 seconds slower than test subjects without dyslexia [12]. A study conducted by Hansen et al., examined the connection between visual deficits in dyslexics and the dorsal stream function[2]. They discovered that dyslexics were less sensitive to coherent motion in dynamic dot displays than a control group. However, the dyslexics did not perform significantly different from the control group when tested in a static environment [13]. The dynamic dot test used in this

[2]The neural stream projected dorsally from the primary visual cortex in to the parietal lobe.

study is the same as used in the application described in this thesis.

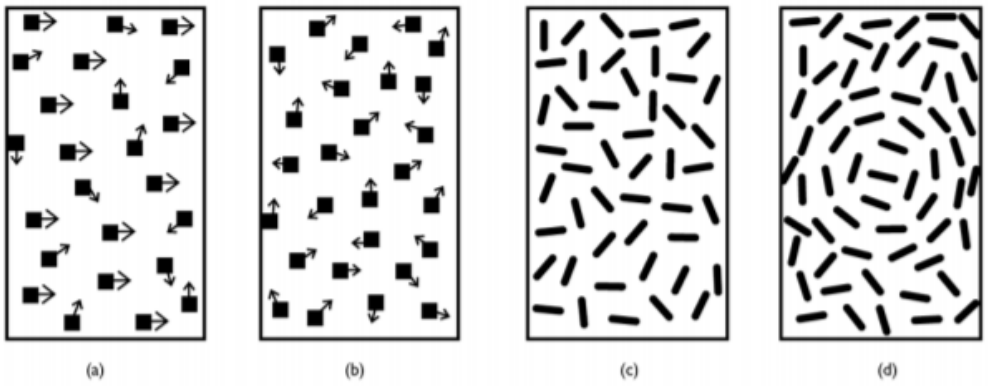


Figure 3.1: Images from the dynamic dot and form tests, testing coherent motion (left) and coherent form (right)

The dot kinematogram showed in figure 3.1 was also used in another study conducted by H. Sigmundsson, P. C. Hansen and J. B. Talcott, where they looked for a connection between developmental "clumsiness" and visual impairment. The results from this research showed that children considered as clumsy had a significantly higher threshold for the visual measures tested. This means that some clumsy children most likely have a visual impairment in addition to their obvious motor problems [14].

3.1.1 Magnocellular System

The dorsal pathway is one part of the brain responsible for visual processing. Within the dorsal pathway, there are cells called magnocellular neurons that are specialised in detecting visual motion. Many dyslexics have reduced activation of the visual areas in the dorsal stream in response to moving targets. This reduced sensitivity to motion indicates reduced sensitivity of the visual magnocellular system in many dyslexics, crucial for visuomotor control. In addition, in the average brain, there is some asymmetry favouring the left part of the brain, especially considering the dorsal pathway. Studies of the brains of known dyslexics show that dyslexics often lack this asymmetry.

Small "brain warts" (ectopias[3]) were also discovered clustered around the temporoparietal junction[4] of the brain in these dyslexics. These warts are associated with widespread disruption of normal neurological connections. The studies done in this field imply that some people with dyslexia also suffer from some kind of visual impairment, especially in relation to coherent motion detection [7]. Yellow filters can improve magnocellular function and improve reading, as magnocells are more sensitive to yellow light. This is exemplified in figure 3.2 and shown in comparison with placebo in figure 3.3.

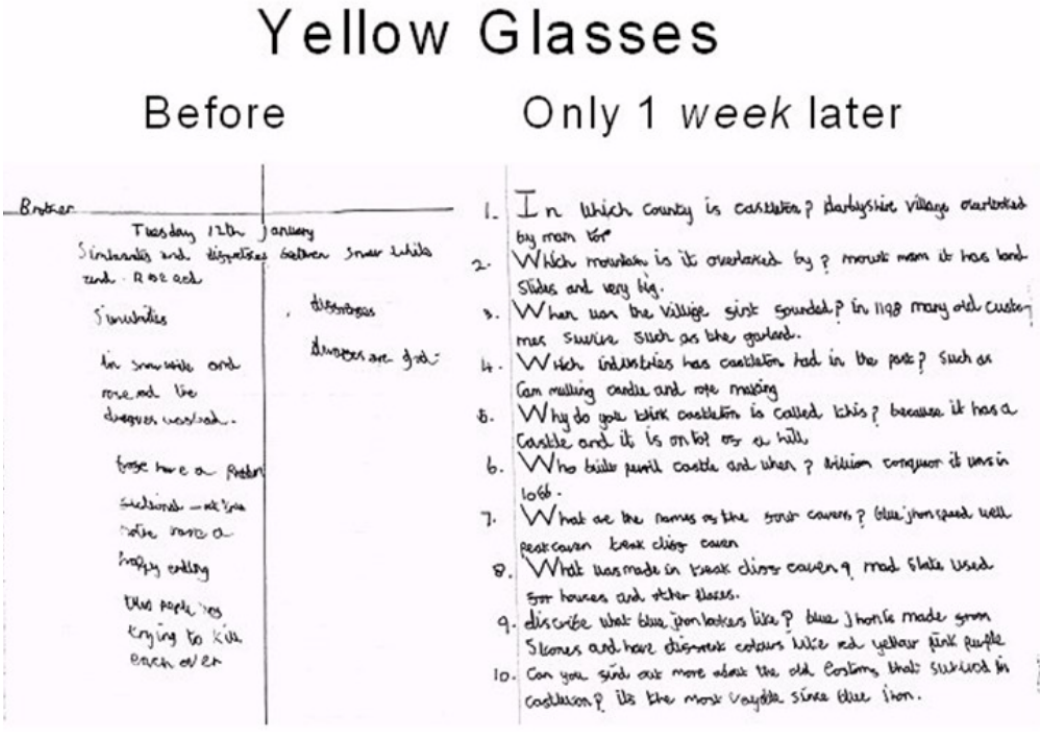


Figure 3.2: Improvement in writing, after using yellow glasses for one week[17].

Blue filters on the other hand, can help the letters keep still when reading [17]. The improvement of literacy when using blue tinted glasses can be seen in figure 3.4.

[3]Malposition of an organ or structure [15]

[4]Relating to the temporal and the parietal bones or regions [16].

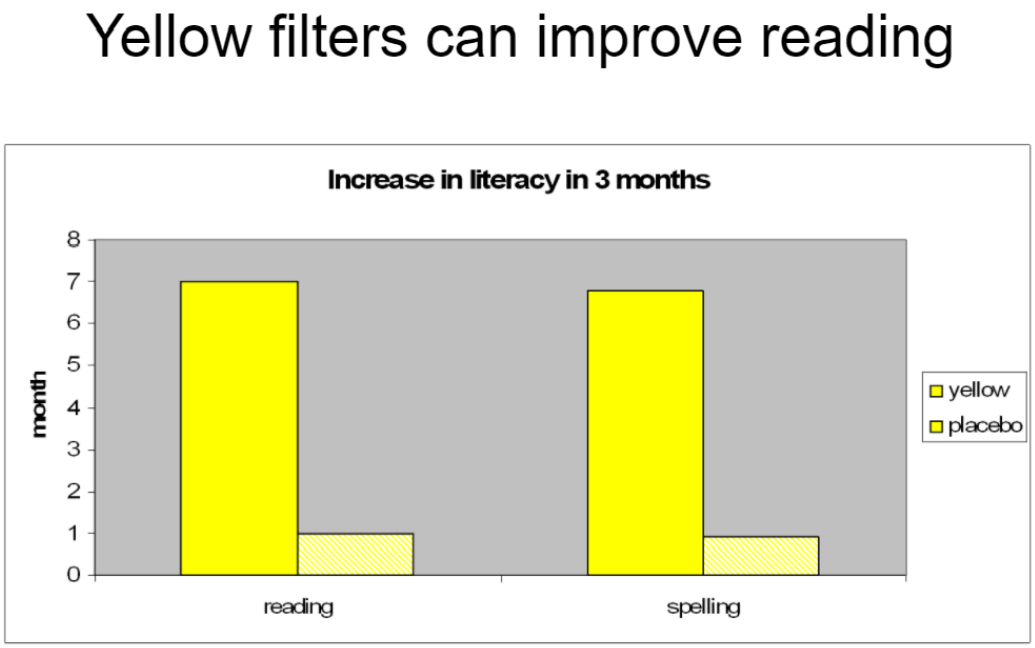


Figure 3.3: Increase in literacy (readin age) after use of yellow glasses [17].

The differences in the brain function explained here might explain the fact that dyslexics score differently than a control group in the various tests described in section 3.2. As many dyslexics have a weakened magnocellular system, it allows for testing of signs of dyslexia by testing people's ability to detect motion and coherent motion. The benefit of this, is that one does not have to know how to read to be tested, and enables dyslexia screening early on in a persons life.

3.2 Detecting Dyslexia Today

There exist many different methods for detecting dyslexia already. The downside of these tests is that many of them rely on the subject already having some reading or writing capabilities. In Norway, dyslexia is normally diagnosed by a specialized doctor, who conducts these tests to ensure that struggles with reading and writing are not caused by some other factor than

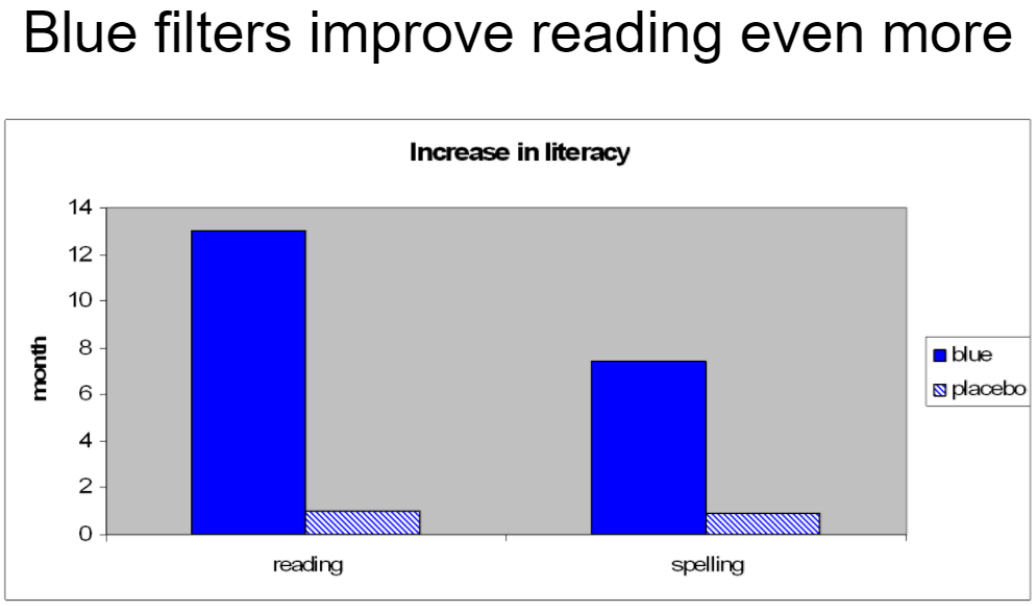


Figure 3.4: Increase in literacy (reading age) after use of blue glasses [17].

dyslexia. Children can start screening in the 2nd grade of primary school [18]. The different tests used to screen for and diagnose dyslexia are often tailored to the subject's age, but one of the most used tests in Norway today, the word chain test (ordkjedetesten), can be performed on all subjects who have some knowledge in reading, regardless of age. The word chain test is a group exercise that is used to map a test subject's ability to decode words. The test consists of 90 words written in the following format: treearrowwhocow sausbeerknifehead. The test subject is to place three lines within each word chain to divide them into four separate words. The test subject is given a score based upon how many word chains he or she is able to divide within a time space of four minutes. The score will be equal to the number of correct answers, with a maximum score of 90 [12]. Test results derived from the application presented in section 3.2.1 have been compared to this word chain test.

3.2.1 App for Early Detection of Dyslexia

During the school year of 2015-2016, an application meant to function as a screening test for detecting dyslexia was developed [3]. The application incorporated the tests shown in figure 3.1. The summer and fall of 2016, H. Sigmundsson and K. S. Egset performed tests with the application on 100 different test subjects; the results were that there is a significant correlation between application test scores and test scores from the word chain test [19]. The goal for the application was to get the tests to function as correctly as possible. Hence, the user interface was secondary to the functionality. The user interface consists of simple white elements on a black background, with colouring to indicate certain functionality. The screen captures below are taken with the default settings specified in the application. The main menu, showed in figure 3.5, consists of buttons in the middle of the screen. You can choose between the three different tests, and settings. The exit button is coloured red to indicate its functionality. Looking at the figure of the main menu, you get a feeling of how the user interface is.

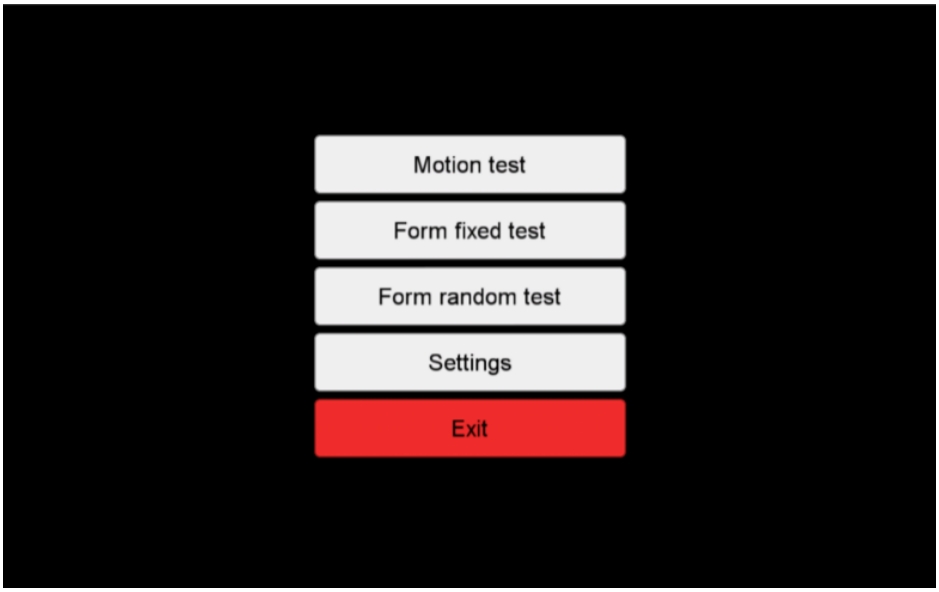
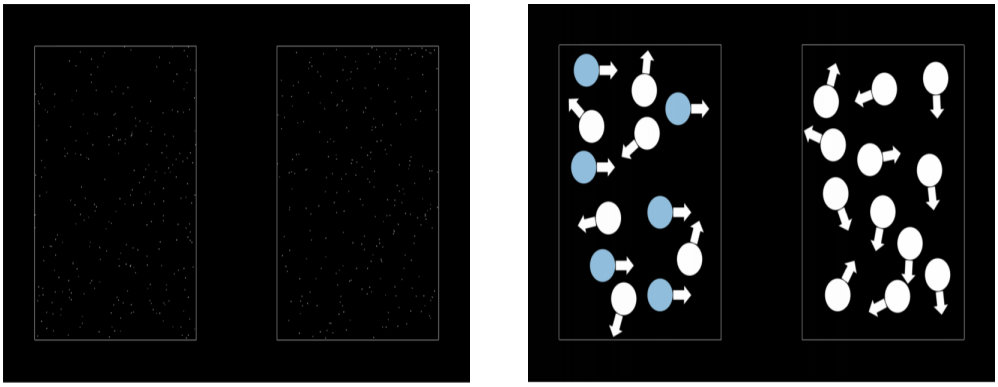


Figure 3.5: Main menu

Figure 3.6a shows a screen capture of the motion test. Each patch contains 300 randomly placed dots with a radius of 1 pixel, and a minimum distance

of 1 pixel between the dots. One of the patches is chosen at random at each interval to contain a coherent motion target. In the chosen patch, a percentage of the dots will move either leftwards or rightwards, reversing every .572 seconds. The rest of the dots will move randomly, changing direction when colliding with other dots after .572 seconds. The test taker should identify the patch with the coherent moving target during the 5 seconds of animation time. After the animation time, the dots will disappear and the test taker can click on the patch they believe contained the coherent motion target. The dots are recalculated with a change in coherency, and subsequently displayed on the screen when the input is registered. Figure 3.6b illustrates the dot animation at 50 percent coherency, where the blue dots are moving coherently to the right. The form fixed auto test is shown in figure 3.6. Each

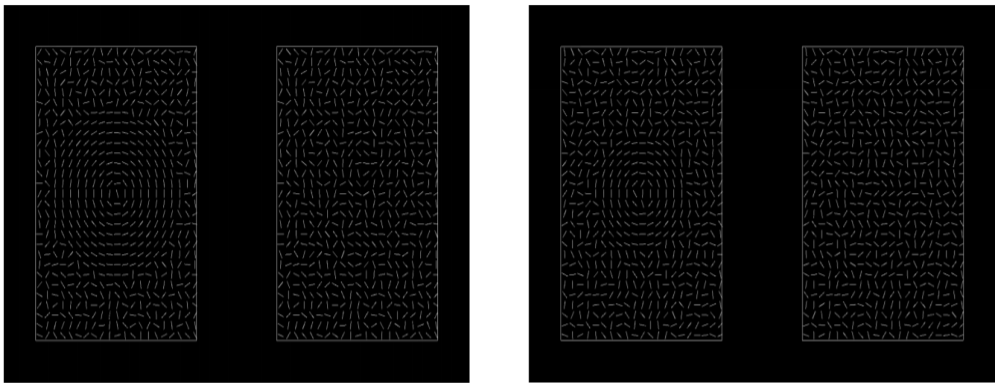


(a) Motion test screen capture.

(b) Illustration of the motion test at 50% coherency.

Figure 3.6: Motion test.

patch consists of 600 line segments, and each of the lines have a length of 0.4° and a height of 1 pixel. One of the patches is chosen at random to contain a percentage of lines that form concentric circles at each interval. The centre of the concentric circles is locked in the centre of the chosen patch. The test taker should search the patches for the concentric circles during an interval of 4 seconds. After the pattern disappears, the test taker can click on the patch they believe contained the circles. As in the motion test, the pattern is recalculated with a change in coherency and subsequently displayed on the screen when the input is registered. Figure 3.7a and 3.7b show the test at respectively 100 and 50 percent coherency.



(a) 100% coherency. (b) 50% coherency.

Figure 3.7: Form fixed auto test.

The form random auto test, shown in figure 3.8, is similar to the form fixed auto test. At each interval, one of the patches are chosen at random to contain a percentage of lines that form concentric circles. The difference is that the centre of the concentric circles will be placed randomly within the chosen patch, with its circumference confined within the patch. Due to the added difficulty of finding the target, the test taker is to search the patches for the concentric circles during the interval of 1000 seconds. The test taker can click the patches at any time.

During the school year of 2016-2017 the Magno-application was upgraded in the master thesis of Johansen and Kirkerød [2]. It was based on improving the usability of the application and improving the user interface. A full overview of the application can be found in appendix A. The screen captures below are the only part of the graphical user interface that needed to be altered. The main menu seen in figure 3.9 that consists of menu to the left with 3 different tests, settings and exit. In the middle of the screen there is a welcome message that provides the short description to the application.

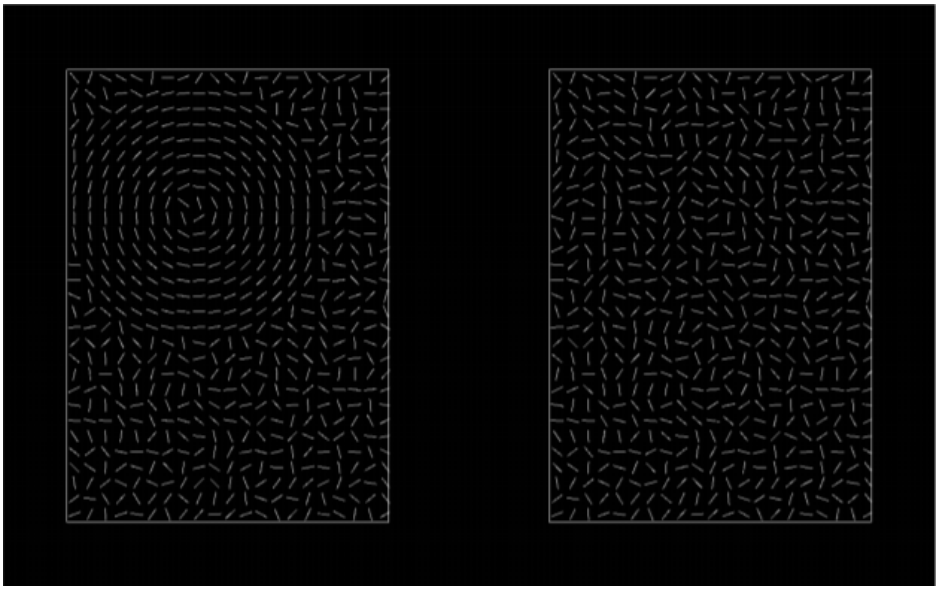


Figure 3.8: Form random auto test at 100% coherency.

Figure 3.9: The new main menu

There are multiple steps for the tutorial, the last step of the tutorial is that the user have to enter their age. Showed in figure 3.10 you will see the header, indicating what kind of test you have chosen, and a text field. Above the text field there is a descriptive text, "Enter your age:", indicating what you should write in the text field. At the bottom a "Start test" button that starts the test. The user can still navigate between the different tutorial screens. If the user chooses to skip the tutorial, or turns off the tutorial in the application settings, the fourth tutorial step is what will be shown.

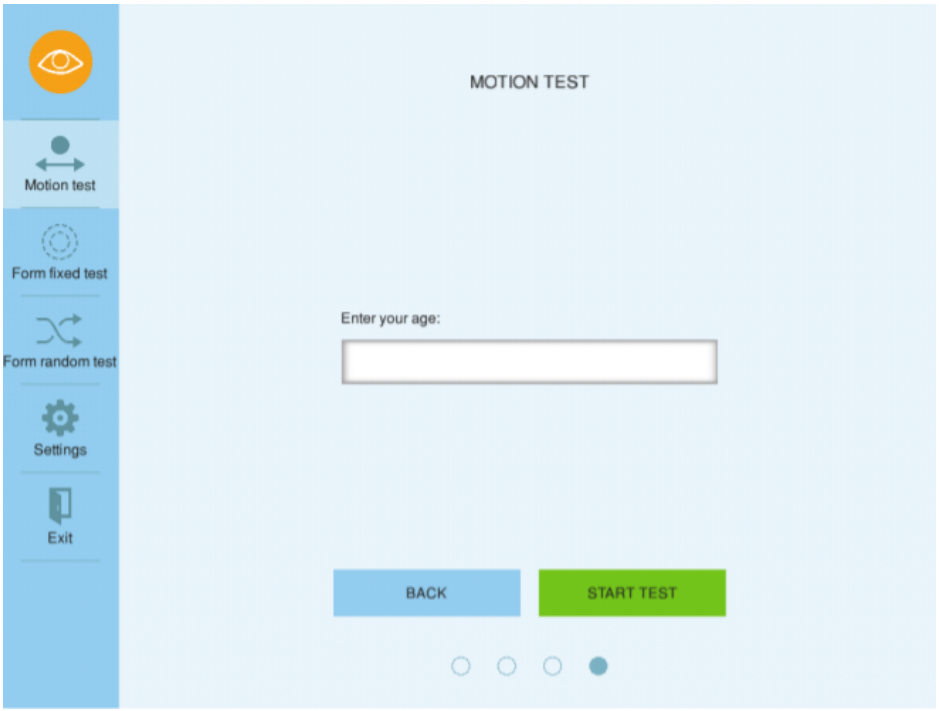


Figure 3.10: Enter age

The content of test result view is the same, regardless of what test the user has chosen. The difference in appearance is which section in the menu bar that is highlighted. This is shown in figure 3.11. The test result view appears when the user is finished with the chosen test. At the top of the screen, there is a header, telling the user that they are viewing the test results. Following, the user sees their score. The score is a value between 1 and 100, representing the final threshold value derived from the completed test. Beneath the score

value, there is a text telling the user whether or not the score is within the normal score range. As there is no available data on what score range is normal for different age groups, the text is static regardless of what age the user enters. Score between 1 and 20 - description "Your score is within the normal score range." Score between 20 and 50 - description "Your score is slightly above the normal score range." Score between 50 and 100 - description "Your score is significantly above the normal score range." In the middle of the screen, there is a progress bar showing a graphical representation of the test score. The progress bar has the number "1" and "100" indicated at each end, and a descriptive text at each extremity. The far left hand side of the progress bar represents low score values, and has the text "Little to no problems". The rightmost side of the progress bar has the descriptive text "Severe problems". There are two buttons at the bottom of the result view. One button will take the user back to the main view of the application. The other button will copy the results along with the test parameters onto the device's clipboard, enabling the user to paste the results into a spreadsheet or a text processing program.

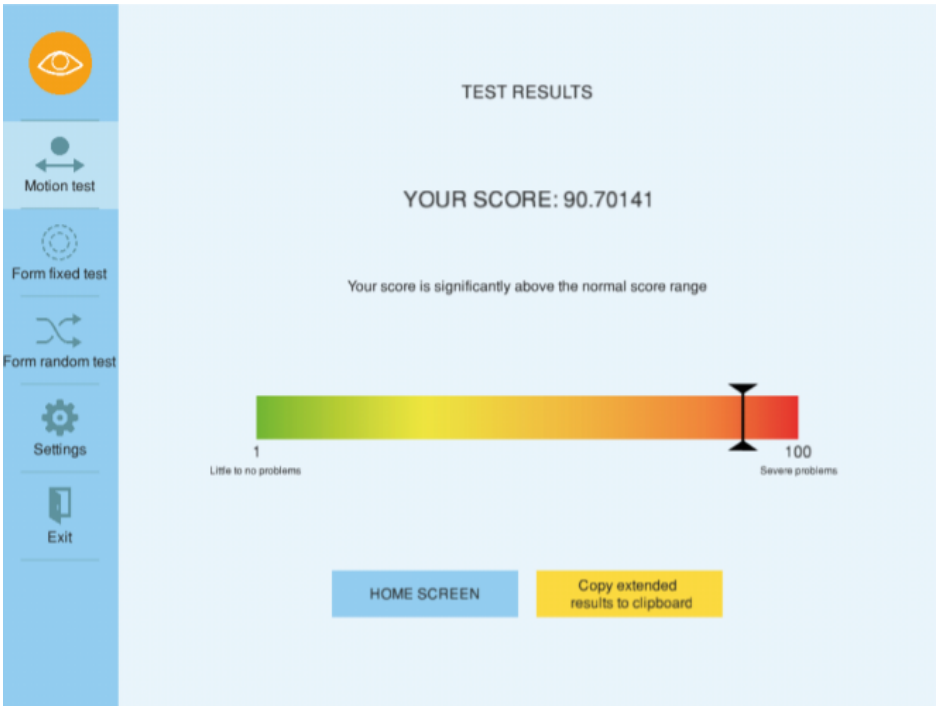


Figure 3.11: Test results

3.3 Requirements

This section will specify and elaborate the requirements for the extended Magno application. The corresponding system-design and implementation of the requirements are detailed in the following two sections. The requirements are derived from the research questions and the vision of prof. Hermundur Sigmundsson, together with the robust system requirements, seen in Table 4-1, established in meetings with Hermundur Sigmundsson. Adding support for the system requirements is important to ensure the continuous support for the project.

ID	Description	Priority
SR1	The application needs to be connected to a database and a system to collect the data to the database.	High
SR2	The system should present the data in such manner that it is easily understood	Medium
SR2.1	The system should only show data to an authorized user	Medium

Table 3.1: System requirements

3.3.1 Functional Requirements

From the very beginning, the advancement of Magno has been motivated by the goal to reduce the complexity of collecting data of early detection of dyslexia. The existing application made it easy for the test subjects, but not for the scientist that want to collect the data. The application Magno, introduced in chapter 3.2, has successfully managed to design a user friendly software, but for collecting and viewing the results of scientific experiments a new part of the system is necessary. In order to support the new part, both some new components and also some changes to the system are needed. The high priority requirements represent the minimum set of requirements for supporting datacollection. The requirements can be seen in Table 3.2. Priorities of the requirements are listed as High (H), Medium (M) or Low (L) in the column labeled 'Pri.'. Inter-dependencies of the requirements are listed in the 'Idep.' column, while the 'SR' column lists the related System Requirements seen in Table 3.1.

ID	Description	Pri.	Idep.	SR
FR1	The database should store the information needed by the stakeholders	High	-	SR1
FR2	The system should connect to the rest-API seamlessly	High	-	SR1
FR3	The rest-API should collect data and input it to the database	High	FR1 FR2	SR1
FR4	The system should be able to store test results for use in comparisons based on age groups and sex	Medium		SR1
FR5	The system should be able to differentiate the different age groups set by the shareholders	Medium	FR2	SR1
FR6	It should be possible to use and navigate the web page with little to none prior introduction	Medium	FR2	SR2
FR7	The web page should be able to show the research information based on a selected age group	Medium	FR2	SR2
FR8	The system should be able to export data to an excel file with some given parameters	Low		

Table 3.2: Functional Requirements

3.3.2 Non-Functional Requirements

Table 3.3 specifies the non-functional requirements for the application. As this research report focuses on introducing new components into an existing application which will use a web-service to collect data, our chosen quality attribute is mainly availability, but also security and scalability.

Priority	Requirement	Requirement
Availability	A1	The system should be available to receive data from the application Magno 99.9% of the day
	A2	The system should be running in less than 20 min after any major incident
	A3	A user should be able to get research data 99.9% of the day
Security	SEC1	The system should only show information to authorized users
	SEC2	The system should not store any personal information about the user
Scalability	S1	The system should be able to handle 50 simultaneous clients
	S2	The administrator should be able to scale-up the system to handle 100 simultaneous clients

Table 3.3: Non-Functional Requirements

Chapter 4

Methods and Implementation

In this chapter I present some of the choices made for making a system of adequate quality, alongside explanations of solutions that were used to achieve the final implementation. Section 4.1 gives an overview over the application overall structure and code structure. In section 4.2 I justify my design choices. Section 4.3 contains a description of the application architecture.

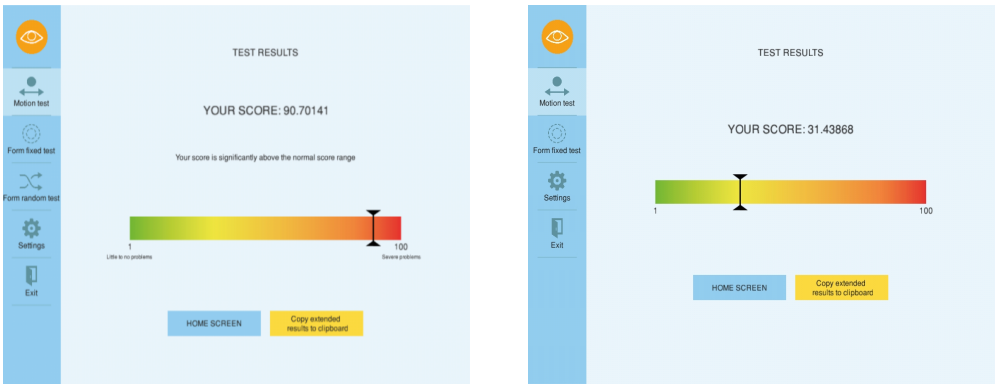
4.1 Application Overview

When implementing the Rest-API, database and web-page of this system, the aim was to make the new parts of the system as non-intrusive as possible, while making necessary changes where they were needed. All screen shots in this section are captured on a Samsung Galaxy Tab A, but as the design is responsive it will look the same on any android device as well as on a desktop. The implementation can be divided into two parts, further development of the design of the application, and converting this design into code.

4.1.1 Magno-app

As the scope for the master thesis this project is built on was usability and creating a good GUI, I did not want to do any more changes than needed. At the start of this project I received some input from Kaja S. Egset about the

GUI of the application, parts of the Result screen was found to be confusing and giving misleading information to the user. In the master thesis of Maja Kirkerød and Thea Hove Johansen I found that the text commenting the score value where set static regardless of age because the application did not have any data to compare the score to [2]. And the names for the progress bar where set a bit extreme so they did not give an accurate representation of the score threshold. These where then removed as showed in figure 4.1a and figure 4.1b.



(a) Before. (b) After.

Figure 4.1: Changes to the test result screen

Sigmundsson et al.[14] found that the form random test did not warrant a good test result in studies concerning dyslexia, so this test would not be used in the studies this application was created to contribute to. This finding resulted in the decision to remove part of the application regarding Form Random Test. Showed on the menu to the left in figure 4.1a.

The last visual change was done as a proof of concept to facilitate more information given the connection to a RestAPI. This gives an option to allow studies conducted on dyslexia regarding sex. And it opens for further development to connect to a identification database with a given ID. The last visual change was added towards the end of the tutorial where the user already input information about his or here age. Showed in figure 4.2

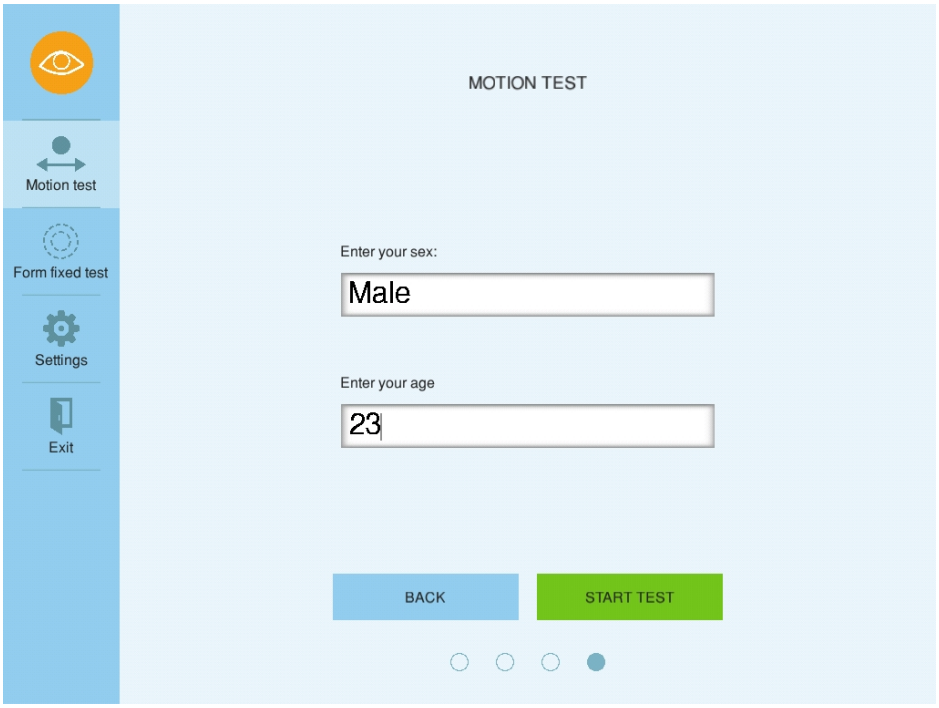


Figure 4.2: Adding sex input to tutorial

4.1.2 Magno Web-page

Considering that the scientist using Magno will not be a computer specialist it was early decided that it was necessary to create a web-page for easy access to the data collected by the application. Since the application has potential to collect privacy information I decided to use the security provided by Microsoft's Asp.Net.

Main page

The home screen seen in figure 4.3 consists of a menu bar on the top side of the screen, containing all the different navigational choices, as well as a welcome message to the web-page. In the top-right side of the page there is a link to the log in page. To gain access to the data-page, one needs to be logged in to an authorized account. The user can click on the Magno logo to get back to the home page from all pages.

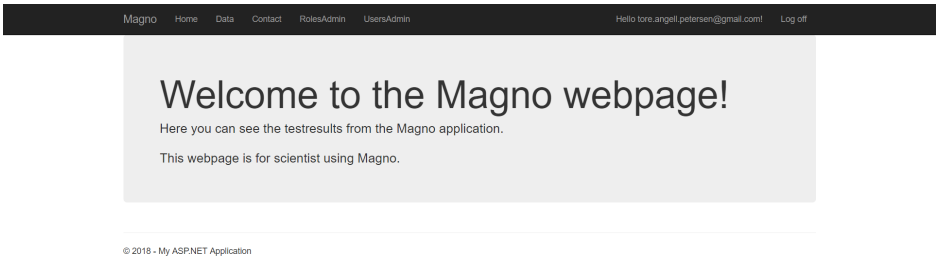


Figure 4.3: Home page

Data-page

The data-page seen in figure 4.4 shows the graphical representation of the scores gathered from the Magno application. This representation can be update by using the update button just above the graph. By using the drop-down menus the user can restrict the age range for the scores shown in the graph.

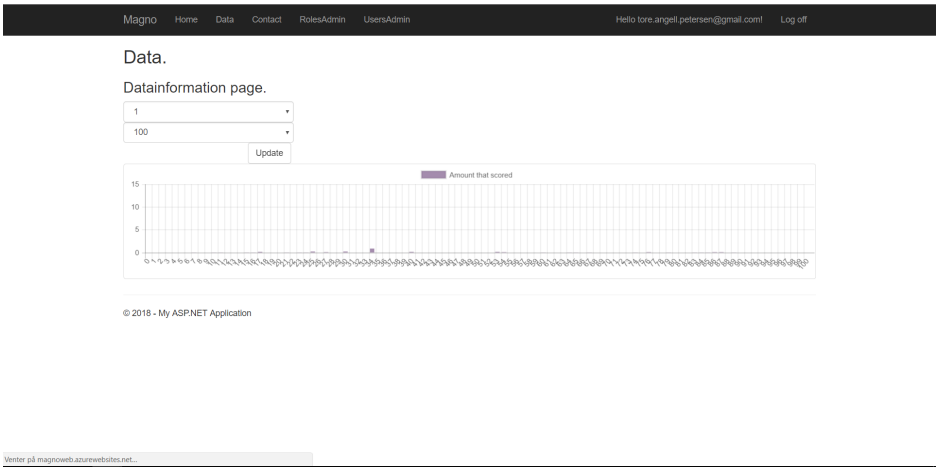


Figure 4.4: Data page

Contact and login

The contact page seen in figure 4.5 gives the contact information for the creator of the web-site for the users to use if any errors are found during use. The log in page seen in figure 4.6 is where the user authenticate them selves to access the data in the data-page.

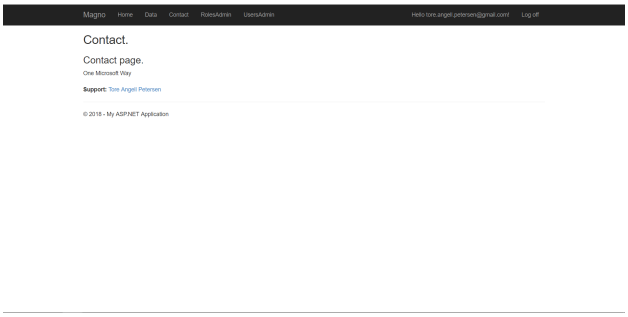


Figure 4.5: Contact page

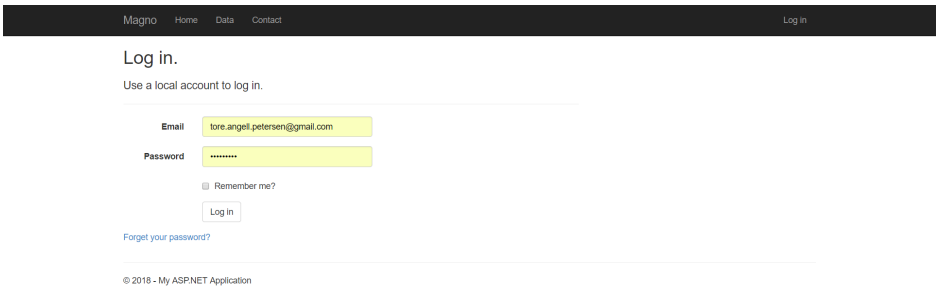


Figure 4.6: Log in page

4.1.3 Database.

ID	Sex	Age	Score	pID	TestType
1	Male	22	17.00	NULL	MOTION TEST
2	Female	27	25.00	NULL	MOTION TEST
3	Male	30	15.00	NULL	MOTION TEST
4	Female	35	16.00	NULL	MOTION TEST
5	Male	45	25.00	NULL	MOTION TEST
6	Female	46	30.00	NULL	MOTION TEST
7	Male	32	34.00	NULL	MOTION TEST

Figure 4.7: Database

The database in this context is relatively basic, since it just collect age, sex, score, testType and potentially ID number. The ID number is not required, even though it was added in the database for the possibility to be compatible with other scientific studies. The layout of the database is shown in figure 4.7. The test-type where added to differentiate between the two different test.

4.1.4 Rest-Api

REST stands for Representational State Transfer. It is a term coined by Roy Fielding in his dissertation [19] to refer a software architectural style. According to [19], an architectural style is a coordinated set of architectural constraints that restricts the roles and features of architectural elements, and the allowed relationships among those elements within any architecture that conforms to the style. In other words, we may regard an architectural style as a set of design patterns with which we can create architectures that exhibit the properties induced by the style. Derived from 12 other related styles, REST is a hybrid (union) style that aims to induce certain architectural properties that are important to distributed hypermedia systems. These properties include usability, simplicity, scalability and extensibility. By introducing these properties with carefully considered trade-offs, REST attempts to minimize latency and network communications, and at the same time, maximizing the independence and scalability of component implementations, including user agent, proxy, gateway, origin server and various connectors, in distributed hypermedia systems.

Node.js

Node.js is a server-side platform built on Google Chrome's JavaScript Engine [20] for easily building fast and scalable network applications. Node.js uses and event-driven, non-blocking I/O model that makes it lightweight and

efficient perfect for data-intensive real-time applications that run across distributed devices. Following are some of the biggest incentives to use Node.js and most important reasons that makes it the first choice of software architects:

- **Asynchronous and event driven** - All API's of Node.js library are asynchronous. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- **Single Threaded but Highly Scalable** - Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers.
- **No Buffering** - Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** - Node.js is released under the MIT license.

4.2 Design Choices

In this section all the design choices that have been made within the application will be described, as well as why they were made.

4.3 Software Architecture

Most of the functionality and all of the GUI for the application was already implemented when I started to work on this system. LibGDX provides a screen adapter class that holds a camera to display the content, and a stage to handle input and the behaviour of actors, like fields and buttons. The software was implemented with the model view controller (MVC) pattern

for the motion and form tests. The classes representing the model of the MVC pattern, keep track of the current state and contain the functionality related to a test [3]. The classes representing the view and controller of the MVC pattern hold the interface and handle input from the user. I had to do changes to the existing code to implement the connection to the server and database. The use of MVC was a big advantage, as this made it possible to implement the server connection without doing changes to the GUI.

4.3.1 MVC

It's reasonable to propose that any given application is likely to change its interface as time goes by, or indeed have several interfaces at any one point in time. Yet the underlying application might well be fairly constant. A banking application that used to sit behind character-based menu systems or command-line interfaces is likely to be the exact same application that today is probably sitting behind a graphical user interface (GUI). Building any particular interface into an application would be to the detriment of both the application, making it less flexible and harder to migrate; and the interface, making it harder to use for other applications.

Model

The model is the unchanging essence of the domain. In object-oriented terms, this will consist of the set of classes which model and support the underlying problem, and which therefore will tend to be stable and as long-lived as the problem itself. The domain model will consist of the objects which represent and support the essence of the problem— Client, Invoice, Booking, ... These are the classes that today's software engineering modelling and implementation would focus on first. Indeed, it is usually considered crucial that the core structure of the solution matches an appropriate and useful structuring of the problem. The domain classes will truly know nothing about the mechanisms that interface them to the outside world.

View

For a given situation there will be one or more interfaces with the model, which we'll call the views (plural). In object-oriented terms, these will consist of sets of classes which give us "windows" (very often actual windows) onto the model, e.g.

- The GUI/widget (graphical user interface) view,
- The CLI (command line interface) view,
- The API (application program interface) view.

Although views are very often graphical, they don't have to be. What will the views know about the model? They have to know of its existence. They must know something of its nature. A bookingDate entry field, for example, might display, and perhaps change, an instance variable of some model class somewhere.

Controller

A controller is an object that lets you manipulate a view. Over-simplifying a bit, the controller handles the input whilst the view handles the output. Controllers have the most knowledge of platforms and operating systems. Views are fairly independent of whether their event come from Microsoft Windows, X Windows or whatever. And, just as the views know their model but the model does not know its views, the controllers knows their views but the view does not know its controller.

4.3.2 Client Multi-Server/Three-tier architecture

Apart from the usual advantages of modular software with well-defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently in response to changes in requirements or technology. For example, a change of operating system in the presentation tier would only affect the user interface code. Typically, the user interface runs on a desktop PC or workstation and uses a standard graphical user interface, functional process logic that may consist of one or

more separate modules running on a workstation or application server, and an DBMS on a database server or mainframe that contains the computer data storage logic.

Three-tier architecture:

Presentation tier
This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents. It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network. In simple terms, it is a layer which users can access directly (such as a web page, or an operating system's GUI).

Application tier (business logic, logic tier, or middle tier)
The logical tier is usually pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing. But in this system the middle tier was only created to collect data from the Magno application and act as the data access layer for the data tier.

Data tier
The data tier is in this system the data persistence mechanisms (database servers, file shares, etc.). Avoiding dependencies on the storage mechanisms allows for updates or changes without the application tier clients being affected by or even aware of the change. As with the separation of any tier, there are costs for implementation and often costs to performance in exchange for improved scalability and maintainability.

4.3.3 Classes

During the implementation, new classes were added, some classes were changed, and in the Magno application most of the classes were left unchanged. The class diagram can be seen in figure 4.8. The classes marked with red colour are new classes, the yellow ones are existing classes where changes were made, and the white ones are classes that have remained the same or have no significant changes. The classes that are most important for

the entire system are described below starting with the changes to Magno application.

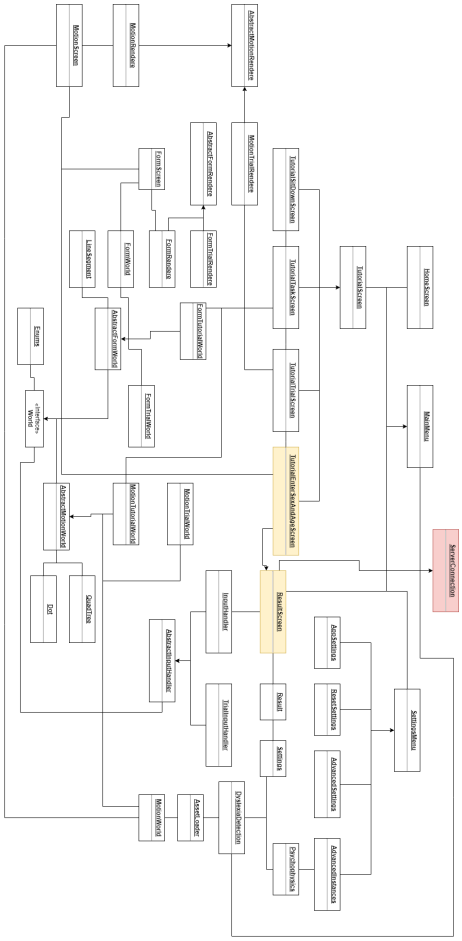


Figure 4.8: Class Diagram for Magno

The TutorialEnterSexAndAgeScreen is the last class in the line of classes abstracted by TutorialScreen. This class has the smallest changes done in the application with just the adding of an textField and an inputField. Then the sex and age are put in a Gdx preference for later uses.

The ResultsScreen class instantiates and builds the content of the result view. The result displayed, both in text and with the score-range bar, is fetched from the Results class, that calculates the threshold score based on the user's

performance during the test. This class also instantiates a server connection class to post data to the RestAPI.

The ServerConnection class builds as the name suggest the connection to the business layer or the server. The class builds a JSON-string that contains the test-subjects age, sex, what kind of test was done and what the score was for the test, which it post to the RestAPI who collects the data to the database. It also has a get method that was made for further development to get the mean score range for the respective age group that the test subject falls under.

The Server class instantiates the Node.js server which handles connection to the database. And it also exposes the server on the port stated in the code.

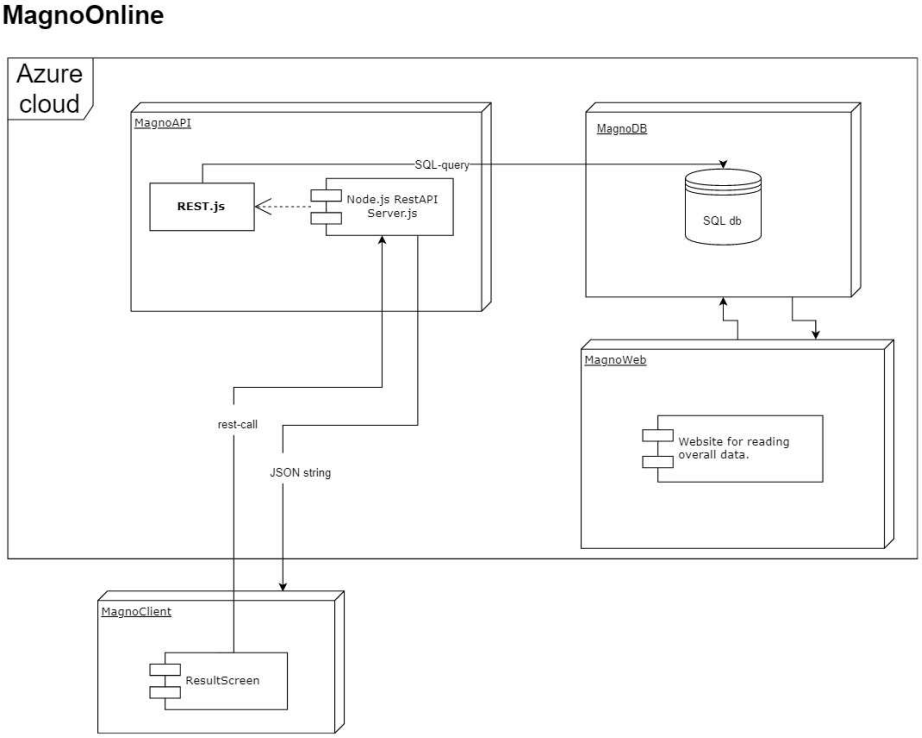


Figure 4.9: Class diagram for the entire system

The Rest class routes the HTTP calls and computes the request if it is either get or push. It is listening on /users for both get and post where get returns all the users and post inserts a new test-subject into the database

with the given data from the JSON-sting. There are three more routes that are listening for get calls on /user/"userID" which return the user with the given userID, /scores that return the sex, age, score and testType for all in the database, and /scoresByAge/"age" that return sex, age, score and testType for all the subjects that are in the age range where the given age falls.

The web page

The web page is created from a template given by Microsoft Visual Studio and it is of the MVC pattern. It was altered to allow security for log-in and using the data page. The web page has email validation system that create an ID-token that is sent to the users email and require the user to have a valid email-address before using the data page. The data page is used to get a graphical view of how the test-subjects did on the test conducted. It is possible to manipulate the age range shown in the graph by using the drop-down menus and the update-button.

Chapter 5

Evaluation

This chapter will evaluated the system in light of the predetermined requirements, found in section 3.3. In section 5.1, a general evaluation is given, and in section 5.2 the technical tools used are evaluated. The requirement fulfillment is evaluated in section 5.3.

5.1 Overall Evaluation

The finished system was not tested before implementation, there were no major changes made to the graphical user interface. The initial plan was to test these changes when they were implemented. The short response time from the server makes the application with database connection not that different than the original application, but due to time constraints the implemented RestAPI and database was not tested either. One can argue that because of the small amount of changes made this is not a major problem. The finished application should still be availability and scalability tested on a larger, more diverse test group before any potential release to the public.

5.2 Evaluation of Technical Tools

This section provides an introduction to the relevant tools and technologies that have been used during this project.

Android Studio

In order to implement the changes to the application Magno, a text editor tool was needed. Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA [21]. On top of IntelliJ's code editor and developer tools, Android Studio offers more features that enhance productivity when building Android apps. Some examples of these features are a flexible Gradle-based build system, a fast and feature-rich emulator, an unified environment where you can develop for all Android devices, and instant Run to push changes to your running app without building a new APK. Android Studio is distributed with templates for creating activities and with the possibility to import samples from the Google Samples repository. Together with the built in terminal it provides a complete package to develop Android applications [**androidStudio**]. Android Studio was chosen as it makes for easy testing on android devices and emulators, as well as being a familiar tool to the author.

Visual studios

In order to implement the server side application, a text editor was needed. Visual Studio is the official Integrated Development Environment (IDE) for development in Microsoft applications. With great build in tools like the debugger where you can quickly find and fix bugs with your systems. And great tools for testing, this IDE fit perfect for me. Visual Studio is distributed with templates for creating activities and with the possibility to import samples from the Microsoft repository. Visual Studio was chosen as it makes for easy connection to the azure cloud and great deployment tools, as well as being a familiar tool to the author[22].

Microsoft SQL Server Management Studios

In order to properly administrate the SQL server on the azure cloud, the clear option was to use Microsoft SQL Server Management Studios(SSMS). SSMS is an integrated environment for managing any SQL infrastructure, from SQL Server to SQL Database. SSMS provides tools to configure, monitor, and administer instances of SQL. SSMS is used to deploy, monitor, and upgrade the data-tier components used by my applications, as well as build queries and scripts. Microsoft SSMS was chosen as it makes for easy management of the database for this application[23].

Postman

In order to properly test out the RestAPI, a tool for API development. Postman makes it easy to test, develop and document API's by allowing developers to quickly put together both simple and complex HTTP requests. Postman is available as both a Google Chrome Packaged App and a Google Chrome in-browser app. With no prior use of a tool like this, Postman was chosen for its good reviews from the community [24].

Azure cloud Solution

In order to implement the server side application, a server solution was needed. Microsoft Azure has a greet reputation from the biggest companies in the world and they still have the little-guy in mind for using their cloud platform. Azure supports a big selection of OS, programming-languages, frameworks, databases and units. With the ease of connecting to Visual studios and amount of support for the solution Azure Cloud was the clear choice for this system [25].

Git

Even though it was only me I decide to use git for easy version control over the project. I quickly chose Git since I had previous experience with it. Git is a free and open source distributed version control system designed to handle

projects with speed and efficiency. It is easy to learn, has a tiny footprint, and fast performance. I found later in the project that you needed git to publish Node.js applications to Azure Git allows and encourages you to have multiple local branches that can be entirely independent of each other [26].

GitHub

I also had prior experience with GitHub. GitHub is a development platform inspired by the way people work. GitHub has built-in review tools that make code review an essential part of a team's process. You can also plan and manage projects from your repositories, create well-maintained docs, and make sure they receive a high level of care [27].

Trello

In order to organize the work flow and keep track of what needed to be done and in what order in this project, a project management tool was needed. The author had previous experience with a tool called Trello. Trello is a web-based project management application that is free, flexible, and a visual way to manage projects and organize anything. It is a cloud based software, and is available both through an application and directly in the web browser [28]. Trello was chosen as the preferred project management tool, as it can assign members to different tasks, and the user can customize the work flow to fit the projects needs. The fact that the author had previous experience with this tool also weighed in heavily on this decision.

ShareLATEX

In earlier projects, I had used ShareLaTeX to write reports. ShareLaTeX is an online collaborative LaTeX editor, with real-time editing incorporated. This allows for easy writing, graphical layout, and compiling, especially with multiple authors. As ShareLaTeX is available online, there is no need for installation [29]. All of these features, along with an incorporated review tool, made me choose to use ShareLaTeX to write and format this thesis.

5.3 Requirement Fulfillment

This section evaluates the functional and non-functional requirements.

5.3.1 Functional Requirement Fulfillment

Table 5.1 evaluates the functional requirements for the application. "ID #" is the requirement number, "Evaluation" describes or evaluates the requirement, while "Fulfillment" describes if the requirement was met or not. The requirements can be found in section 3.3.

ID	Evaluation	Fulfillment
FR1	The database stores information about sex, age, score and what type of test was taken.	Attained
FR2	The system connects to the RestAPI without disturbing the user taking the test	Attained
FR3	The rest-API does collect data from the application and inputs it into the database	Attained
FR4	The system are able to store test result for use in comparisons by collecting the age, sex and score of a test-subject	Attained
FR5	The system are able to differentiate the different age groups by a set mapping table as set by the shareholders	Attained
FR6	I did not have the time to test if a user would be able to navigate the web page with little to none introduction	Not Attained
FR7	The web page show research information based on the selected age groups that are given in the drop-down menu on the data page	Attained
FR8	The system are not able to export data to an excel file from the web page due to some unexpected problem with the implementation.	Not Attained

Table 5.1: Functional Requirements

FR1, FR2, FR3, FR4, FR5 and FR7 have all been attained. FR6 and FR8 are not attained. The web page was not tested by anyone other than the developer, so I can not say with certainty if it is easy to use for other user. Using an SQL management tool this data is easily collected, though this is not an adequately solutions given that the users of this web page is not computer specialists. With the given time-frame and the addition of the FR6 and FR8 has a low priority in the project this requirement was not solved. 6 out of 8 requirements have been met, where the one requirement that were not met had a low priority. This is considered high requirement fulfillment.

5.3.2 Non-Functional Requirement Fulfillment

Table 5.2 evaluates the non-functional requirements listed in chapter 3.3.

Requirement	**Evaluation**	**Fulfillment**
A1	This was not tested, so it is no way to be sure that the data from the application Magno will available all of the time, even with health checks on the application	Not Attained
A2	The azure cloud solution is set up to give the system administrator a message when any major incident happens with the application by mail. This has not been tested adequately	Not Attained
A3	With the health check and message to the system administrator the system should be available, but the application has not been tested adequately so this need to be tested	Not Attained
SEC1	The system only show data to personnel with an authorized users created by the system administrator	Attained

Requirement	Evaluation	Fulfillment
SEC2	The system does not store any personal information about the user at this time, the database is built to store some personal ID for further use to connect this test data to any other scientific studies, at this time the database only fill this with the value of "null"	Attained
S1	I was not able to test if the system are able to handle 50 simultaneous clients as it is today.	Not Attained
S2	I have not tested the scaling of the system, but by using the azure cloud solution together with Node.js the system is built to scale.	Not Attained

Table 5.2: Non-Functional Requirements

Only two of the non-functional requirements have been attained. The rest of the requirements have been tested adequately. So I can not say that they have been attained when I do not know the outcome of the tests. The scalability requirement was not prioritized with the knowledge that node.js and azure cloud in combination makes an scalable system. The application is built to always be available and, secure and scalable enough, based on the standard decided prior to the project start. Requirements need to be tested before the application is taken for use.

Chapter 6

Discussion, Conclusion, and Further Work

In this chapter I present a discussion around the project, along with a conclusion and further work that needs to be done. In section 6.1, the results from the evaluation, and the project development in general, are discussed. In section 6.2, the research questions are answered, and in the last section, section 6.3, alternatives to further work are presented.

6.1 Discussion

This project began as a continuation of a previous master thesis, viewed in section 3.2.1. The theoretical research and the implementation of the precursor to this project was thoroughly conducted. Still, in order to be able to use this application for a larger scientific studies, it is important that the application not only work as its own entity, but also has the means to collect the data needed to conduct the studies. Without a proper data collection part of the entire system, the functionality is not conveyed to its full extent, and usages for the application decreases. The finished system design has its shortcoming, based on the fact that it has not been sufficiently tested, especially for a target group with greater diversity. There have been made changes that can not be corroborated as to whether they improve on the system or not, even though I have sufficient experience with web-services to

say that the changes most likely will have a positive impact. It is still safe to say that the current system, developed during this project, provides for a greater area of application than the previous system. The iterative process of creating an entire system to collect information from an existing application and show it to, is time consuming. Also to create a data collection system in about 6 months, with the proper testing and implementation needed, is probably not an optimal time estimation. That being said, the process has been exceedingly educational in regards to several aspects; how to quickly familiarize myself with new technologies, the different aspects of dyslexia, and responsive design in applications. When coming across a problem, the ability to work around the issue and find solutions has been vital to the project. Encountering obstacles increased both learning and motivation for me. This challenged my expertise and current knowledge, forcing me to look beyond my existing knowledge space and acquiring new experiences and further understanding of a subject.

6.2 Conclusion

The objectives of this project were to further develop and implement a data collection system. Research questions were made to help in steering the research and development in the correct direction. The data collection centred development process was used to develop the RestAPI and database. The documentation found on these subject were analyzed to help me answer the research questions that have guided the project. The conclusion to this project will review these research questions and determine whether or not they have been adequately answered.

RQ-1 What are the problem areas with developing the web-service in this project?

There were found several problem areas with the system developed in this project. The first was that this project had a difficult scope to define and the requirements are often hard to clearly define. This because my specialization project did not build up to this specific project. It was clear that the requirements were not perfect. The main problem is that to often the

requirements are to broad, in this case some of the requirements was not met because of other requirements were to broadly scoped. The solution ended up using technology that the author was not familiar with. So this made it a bit harder to meet all the requirement.

RQ-2 What are the problem areas with collecting personal data?

There are several problem areas with collecting personal data. With the new update to General Data Protection Regulation (GDPR) we have some new regulations to consider when building this application. Privacy by design as a concept has existed for years now, but it is only just becoming part of a legal requirement with the GDPR. At it's core, privacy by design calls for the inclusion of data protection from the onset of the designing of systems, rather than an addition. More specifically system manager shall implement appropriate technical and organizational measures in an effective way in order to meet the requirements of this Regulation and protect the rights of data subjects. GDPR calls for system manager to hold and process only the data absolutely necessary for the completion of its duties (data minimization), as well as limiting the access to personal data to those needing to act out the processing [30]. Given this I decided to implement the database and RestAPI code to collect the data needed to connect the test-subject to other tests, but did not implement the code in the Magno application to collect any means identification. Also I decided to only show test scores based on age in the graph on the web page.

RQ-3 How can we compliment existing code with the implementation of the new web-service?

In order to compliment the existing code, I reviewed the frameworks and libraries previously used. This helped me to understand which parts of the code belonged to the frontend. As my task description encompasses backend development, I could easily understand which parts of the code to change and not. Since the last master thesis where all about usability, it was early decided that connecting to a web-service should not interfere with the rest of the application. By using Node.js and Azure as the web-service I was able to manage low response time that would not interfere with the GUI. It would

require some changes to the GUI if we decided to add the mean values for the result screen on the threshold bar, but given that this is already implemented on the web-service I expect that this would not create any big delays for the GUI other than rendering the information to the result screen.

RQ-3.1 What technical tools are best suited to aid in implementing a web-service for data collection, based on previously implemented functionality?

With Node.js asynchronous and event driven there is not going to non-blocking I/O model that makes it lightweight and efficient. Node.js is using Google Chrome's V8 JavaScript Engine which makes it really fast [20]. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Given that Azure is build to accommodate any programming language it has already build in tools to develop and deploy Node.js applications with ease. And the fact that Azure cloud give a really easy database management makes the combination of these two tools the best way to solve this project.

6.2.1 Final Conclusion

In my opinion, the application sufficiently answers the research questions and fulfills the task description. The system have made it easier for scientist to collect data for scientific studies. There will always be areas for improvement, but our thesis manages to address the majority of elements specified prior to the project initiation, in the form of task description, research questions, and requirements. The application delivered along with this thesis can be taken into use as is, though I would recommend further development.

6.3 Further Work

This project has had its main focus on data collection development and implementation. While the web-service is finished, there are still some aspects

that can be further developed before deploying this application to the masses. Some key elements for further work are as follows:

- Implementation of non-acquired requirements.
- Final scalability testing and fine tuning of the performance.
- Implementation of functionality enabling selection of language.
- Implementation of feedback to the user based on processed information, such as age and/or test score.

These elements are constructed of elements found in this project and some of the elements from the previous master projects[2]. When all these elements have been considered and potentially implemented, one can consider revisiting and revising the requirements. One might see it necessary to add or subtract further functionality, and means of distribution must also be considered. Specifying the environments in which the application is to be used is key, and narrowing down the target audience. It is also important that the application tests are thoroughly documented and tested in order to validate any data delivered by the application.

Appendix A

The application

Here I will show you the entire application using screenshots.

Figure A.1: Main menu

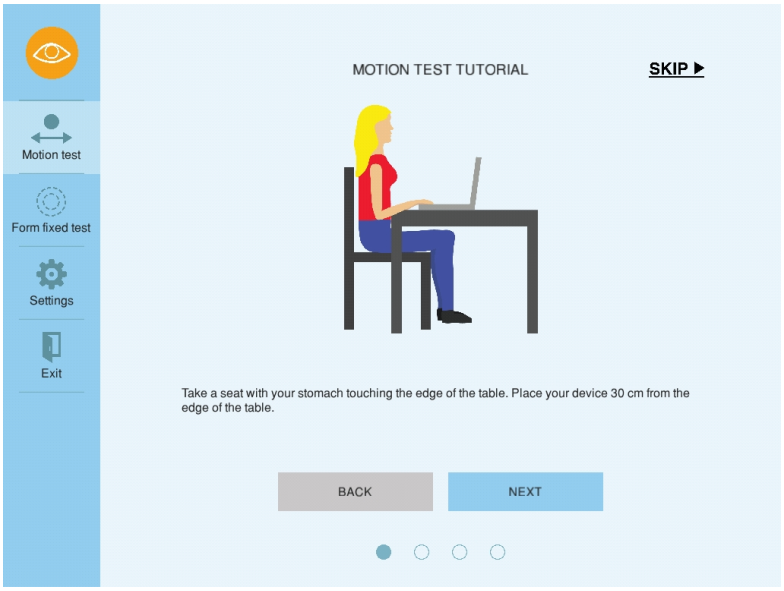


Figure A.2: First tutorial page

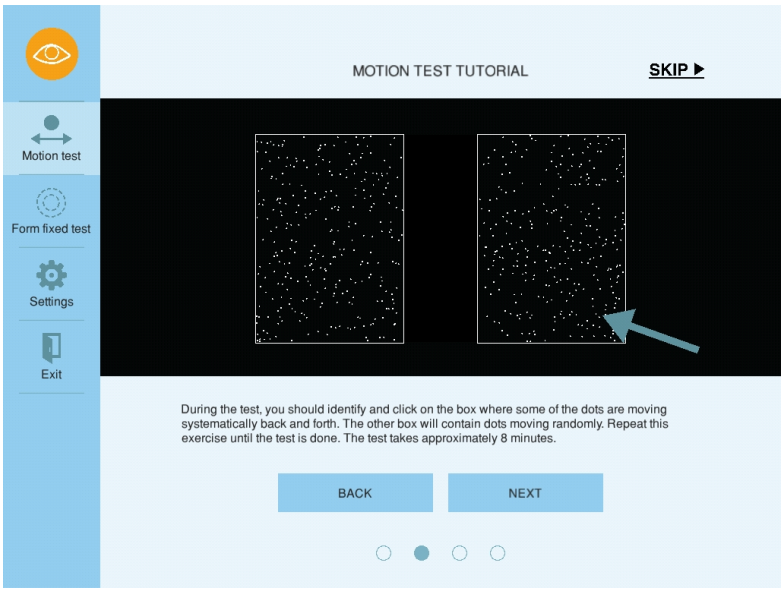


Figure A.3: Second tutorial page

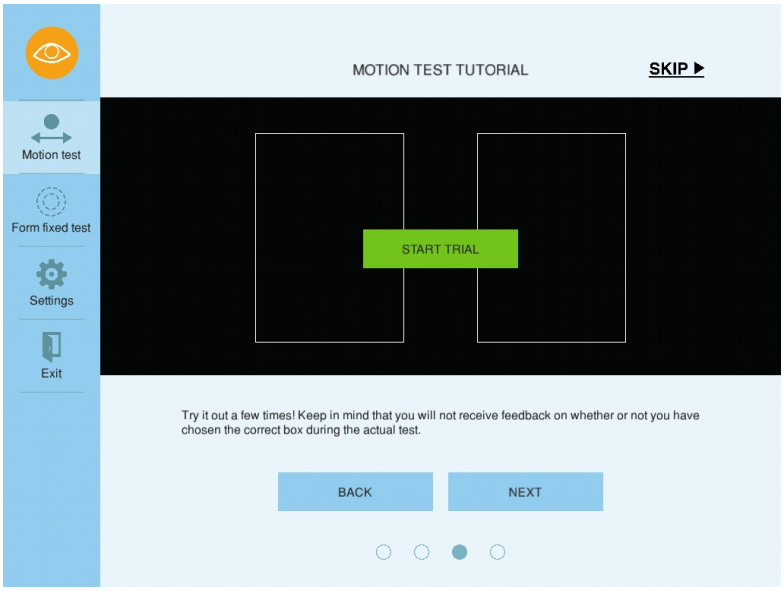


Figure A.4: Tutorial test start page

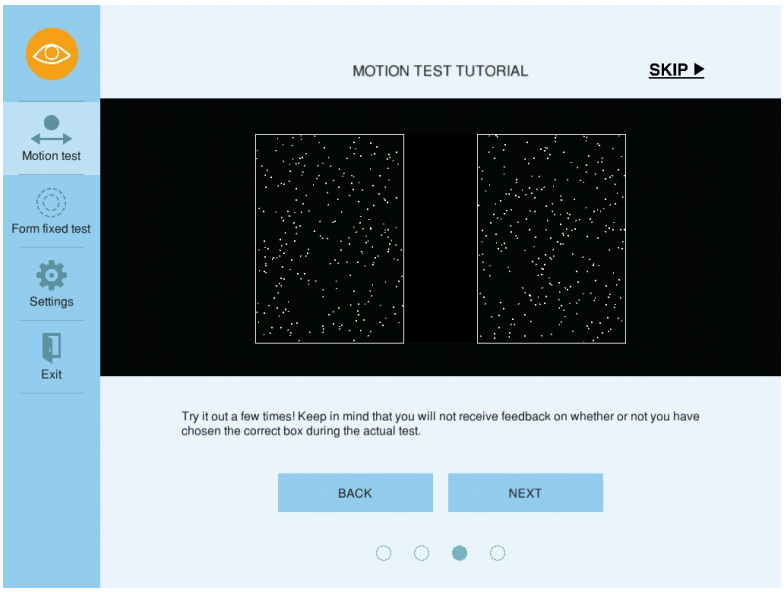


Figure A.5: Tutorial test page

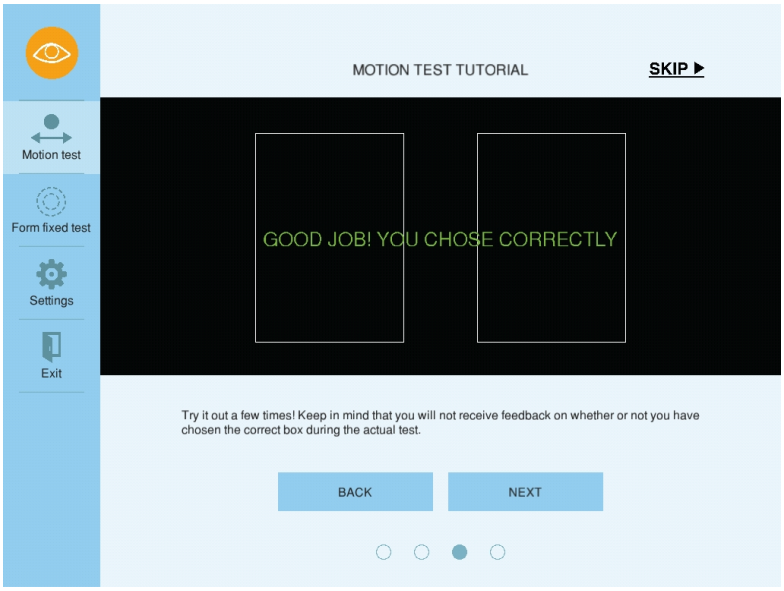


Figure A.6: Tutorial test response when you choose the correct box

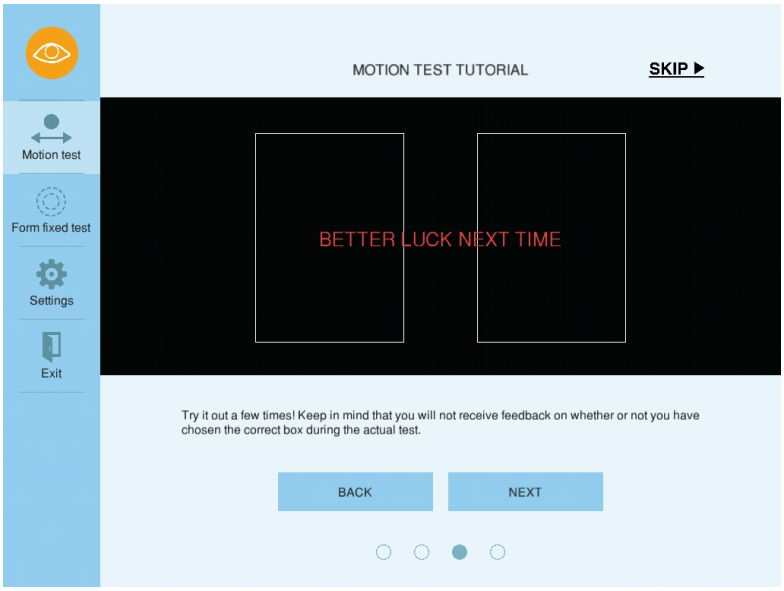


Figure A.7: Tutorial test response when you choose the wrong box

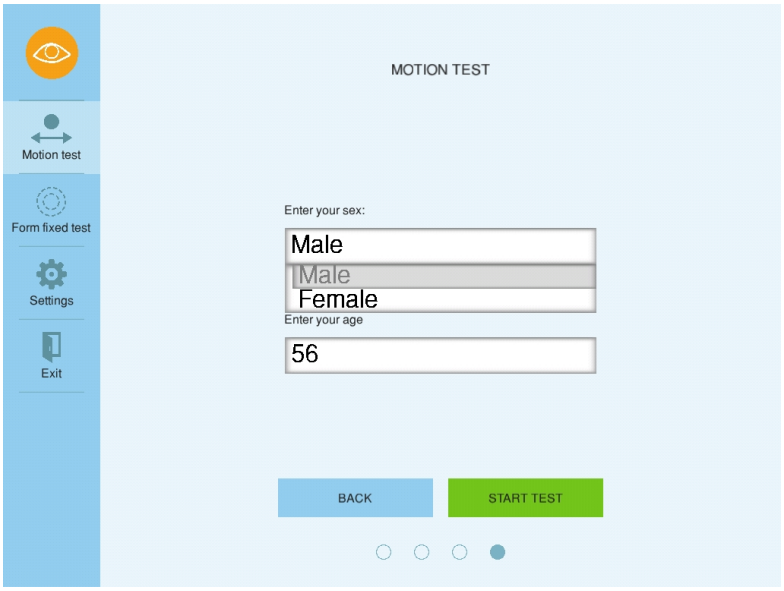


Figure A.8: Last tutorial page with the drop down box for choosing your sex

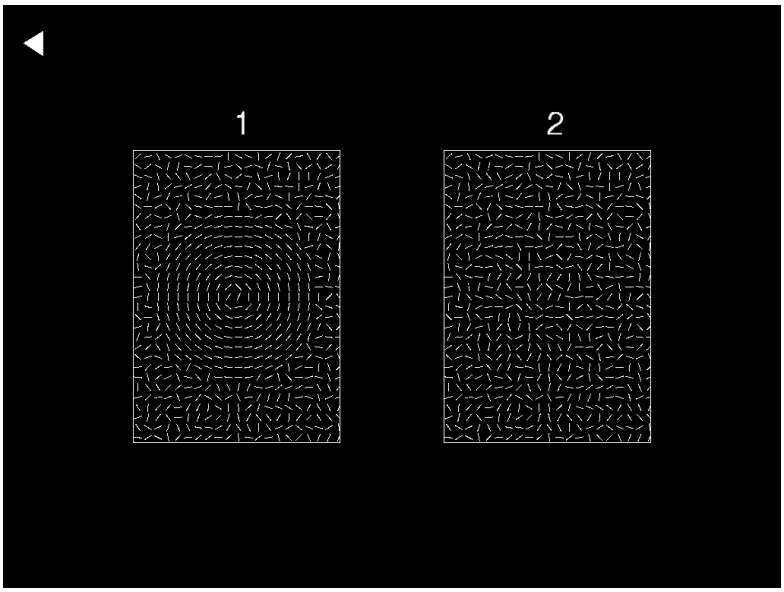


Figure A.9: Form fixed test screen with full circle

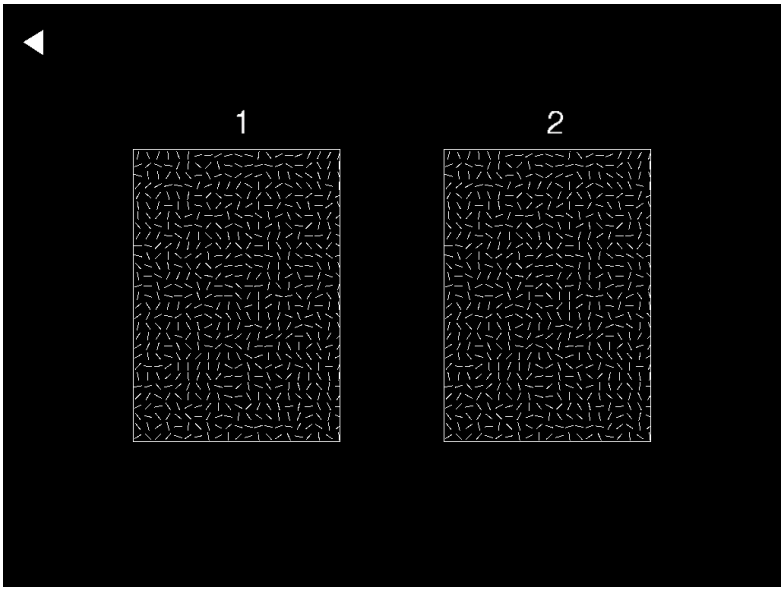


Figure A.10: Form fixed test screen with almost no circle

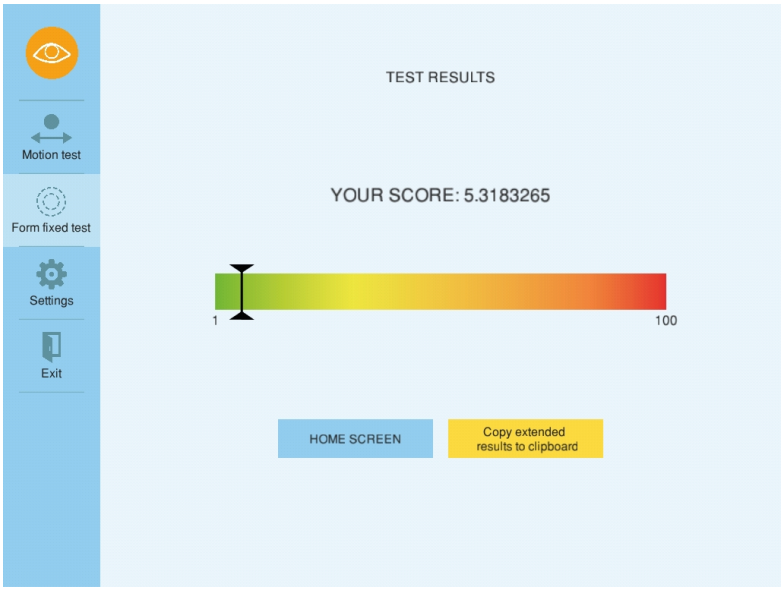


Figure A.11: Form fixed test result

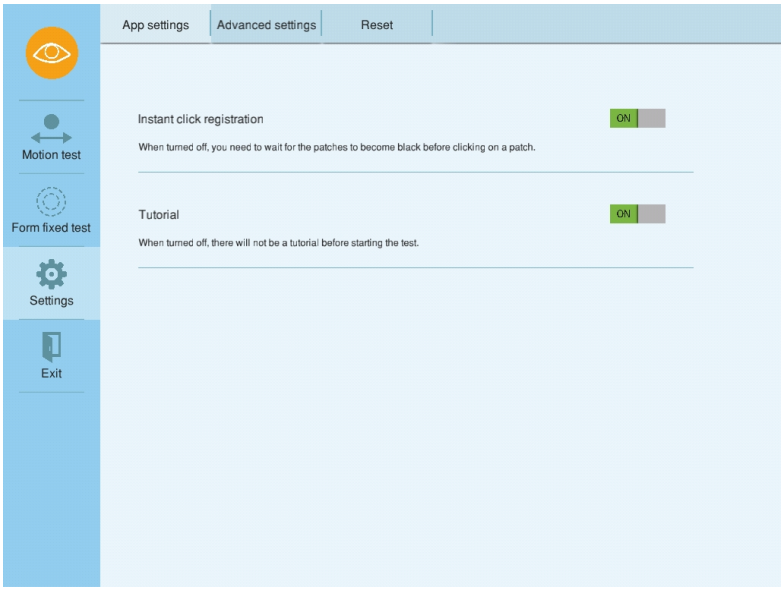


Figure A.12: App settings

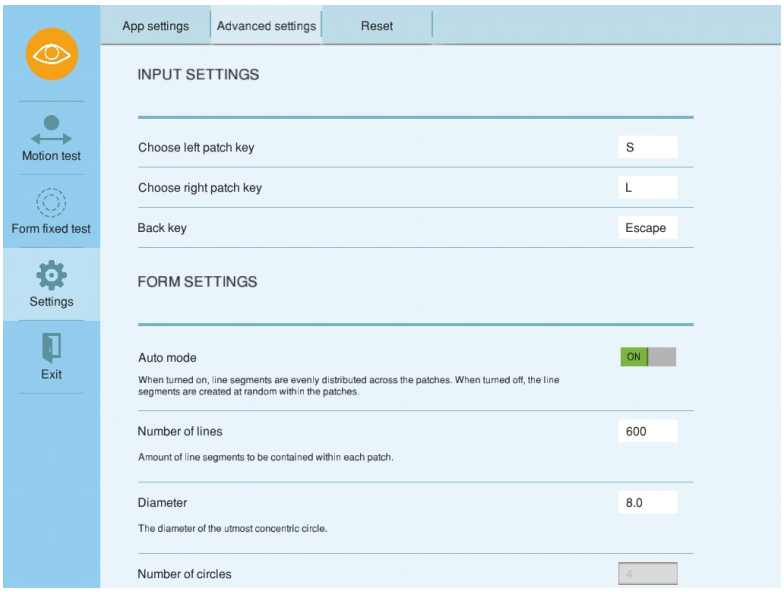


Figure A.13: First part of Advanced settings

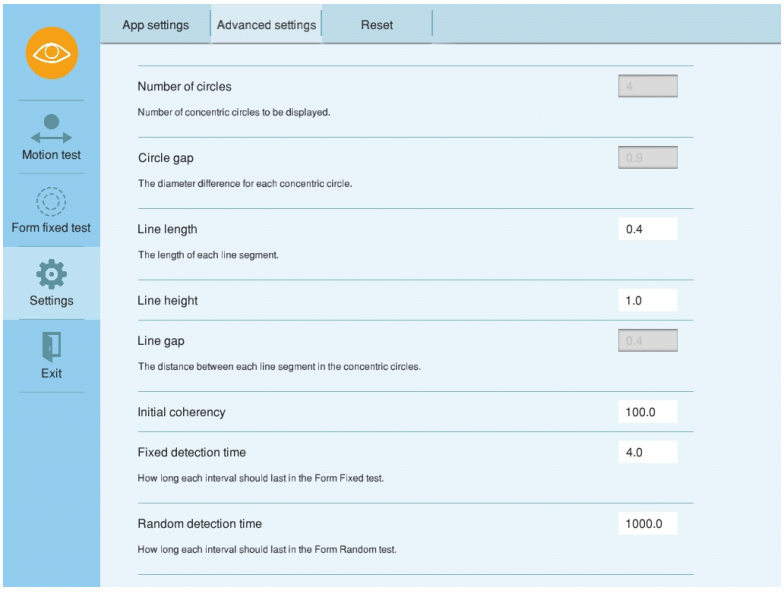


Figure A.14: Second part of Advanced settings

Figure A.15: Third part of Advanced settings

Figure A.16: Fourth part of Advanced settings

Figure A.17: Fifth part of Advanced settings

Figure A.18: Sixth part of Advanced settings

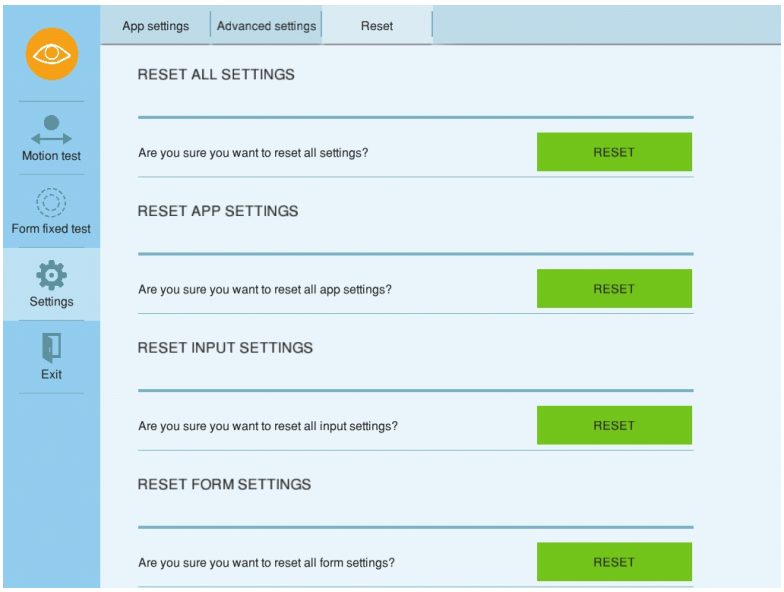


Figure A.19: First part of reset settings

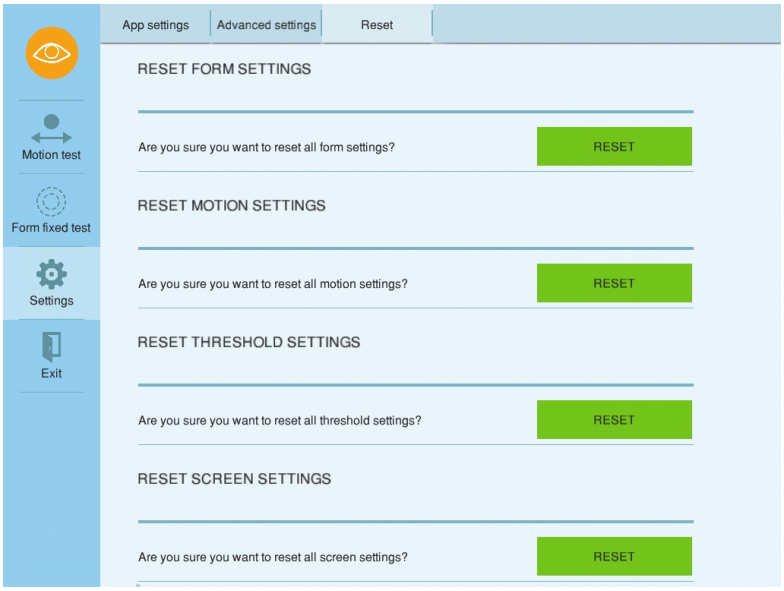


Figure A.20: Second part of reset settings

Bibliography

[1] Sally Shaywitz. *What is Dyslexia?* URL: http://dyslexia.yale.edu/dyslexia/what-is-dyslexia/.

[2] Thea Hove Johansen and Maja Kirkerød. "Magno: An Application for Detection of Dyslexia". In: *MA thesis* (2017).

[3] Bjørnar Håkenstad Wold. "App for Early Detection of Dyslexia". In: *MA thesis* (2016).

[4] Maja Kirkerød and Thea Hove Johansen. "Designing an Application for Detection of Dyslexia". In: *Specialisation project* (2016).

[5] Design Science Research in Information Systems. "Hevner, A., & Chatterjee, S." In: *Design Research in Information Systems* (2010), pp. 9–22.

[6] Briony J. Oates. "Researching information systems and computing". In: (2005).

[7] John Stein. "The magnocellular theory of developmental dyslexia". In: *Dyslexia* 7.1 (2001), pp. 12–36. URL: http://dx.doi.org/10.1002/dys.186.

[8] IDA Board of directors. *Definition of Dyslexia*. 2002. URL: https://dyslexiaida.org/definition-of-dyslexia/.

[9] British Dyslexia Association. *Dyslexia Definitions*. URL: http://www.bdadyslexia.org.uk/dyslexic/definitions.

[10] Oxford English Dictionary. *Dyslexia*. URL: https://en.oxforddictionaries.com/definition/dyslexia.

[11] Dyslexia Research Trust. *Genetics of Dyslexia*. URL: http://www.dyslexic.org.uk/research/genetics-dyslexia.

[12] H. Sigmundsson. "Do visual processing deficits cause problem on response time task for dyslexics?" In: *Brain and cognition* 58.2 (2005), pp. 213–216.

[13] P. C Hansen et al. "Are dyslexics' visual deficits limited to measures of dorsal stream function?" In: *Neuroreport* 12.7 (2001), pp. 1527–1530.
[14] H. Sigmundsson, P. C. Hansen, and J. B. Talcott. "Do 'clumsy' children have visual deficits". In: *Behaviournal Brain Research* 139.1 (2003), pp. 123–129.
[15] The Free Dictionary Medical Dictionary. *Ectopia*. URL: http://medical-dictionary.thefreedictionary.com/Ectopia.
[16] The Free Dictionary Medical Dictionary. *Temporoparietal*. URL: http://medical-dictionary.thefreedictionary.com/Temporoparietal.
[17] John Stein. "The Current Status of the Magnocellular Theory of Dyslexia". 2014. URL: http://slideplayer.com/slide/6120/.
[18] Dysleksi Norge. *Avdekke dysleksi*. URL: http://dysleksinorge.no/fagstoff/.
[19] Roy Thomas Fielding. *Architectural Styles and the Design of Network-Based Software Architectures, Ph.D. Dissertation*. 2000. URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
[20] Node.js Foundation. *Node.js*. URL: https://nodejs.org/en/.
[21] JetBrains. *IntelliJ IDEA*. URL: https://www.jetbrains.com/idea/.
[22] Microsoft. *Visual Studio IDE*. URL: https://www.visualstudio.com/vs/.
[23] Microsoft. *SQL Server Management Studio (SSMS)*. URL: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017.
[24] Postman. *Postman Makes API Development Simple*. URL: https://www.getpostman.com/.
[25] Microsoft. *Din visjon, dine resultater, din sky*. URL: https://azure.microsoft.com/nb-no/.
[26] git. *git –fast-version-control*. URL: https://git-scm.com/.
[27] GitHub. *Built for Developers*. URL: https://github.com/.
[28] Inc Trello. *Trello*. URL: https://trello.com%20(.
[29] Henry Oswald and James Allen. *ShareLaTeX*. URL: https://github.com/sharelatex/sharelatex.
[30] European Commission. *2018 reform of EU data protection rules*. URL: https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en.