



Western Norway
University of
Applied Sciences

BACHELOR'S THESIS

Psychomotor Vigilance Test Application Development

Oskar Midbøe, Sindre Steinsvik and Eirik Tømmervik

Bachelor Report, Computer Engineering

Department of Computing, Mathematics and Physics

Faculty of Engineering and Science

Supervisor: Ilona Heldal

Submission: 02.06.19

Number of words: 12 642

I confirm that the work is self-prepared and that references/source references to all sources used in the work are provided, cf. Regulation relating to academic studies and examinations at the Western Norway University of Applied Sciences (HVL), § 10.

TITLE PAGE FOR MAIN PROJECT

<i>Report title:</i> Psychomotor Vigilance Test Application Development	<i>Date:</i> 02.06-19
<i>Author(s):</i> Oskar Midbøe, Sindre Steinsvik, Eirik Tømmervik	<i>Number of pages, w/o attachments:</i> 41
	<i>Number of pages, attachments:</i> 8
<i>Education Program:</i> Bachelor, Computer Science	<i>Git-hub directory:</i> Not public
<i>Contact at faculty:</i> Ilona Heldal	<i>Confidentiality:</i> None
<i>Remarks:</i> -	

<i>External company:</i> Western Norway University of Applied Sciences, Haugesund	<i>External company reference:</i> -
<i>External contact:</i> Margareta Holtensdotter Lützhöft	<i>Contact:</i> Margareta.Holtensdotter.Lutzhof@hvl.no

<i>Summary:</i> This bachelor thesis covers the development of a cross-platform application for Android and iOS, written in Xamarin. It is accompanied by a Windows Presentation Foundation application and connected by an Azure server containing a Web-API and MySQL database. The application is a psychomotor vigilance test (PVT), to be used for research. The development process utilizes Scrum and Human Centred Design (HCD). Usability testing was performed to test the application design and user experience(UX).
--

Key words:

Cross platform	PVT	Human Centred Design
----------------	-----	----------------------

PREFACE

The following bachelor thesis is written by three students from Western Norway University of Applied Sciences. Oskar Midbøe and Sindre Steinsvik from Information Technology, and Eirik Tømmervik from Computer Engineering. The project was based on an idea by Professor Margareta Holtensdotter Lützhöft, from HVL Haugesund, Dept. of Maritime Studies.

First and foremost we would like to thank Amanda Søreide, for the two most important clicks of a mouse during our entire project.

We would like to thank Remy Andre Monsen, for helping us with servers, databases and for letting us use his virtual machines.

Thank you to Thomas Reite and Magnus Marthinsen, for your wisdom and guidance.

We would like to extend a large thank you to Volker Stolz, for his invaluable help, getting us across the finish line. We owe you big time.

Our project employer, Margareta Holtensdotter Lützhöft, has been a pleasure to work with, and we want to thank her for all her time and feedback throughout the project.

Lastly, we would like to thank our project supervisor, Ilona Heldal.

Table of Contents

PREFACE	3
1 INTRODUCTION	1
1.1 GOAL AND MOTIVATION	1
1.2 CONTEXT	2
1.3 LIMITATIONS	2
1.4 RESOURCES	3
1.5 ORGANIZATION OF THE REPORT	4
2 PROJECT DESCRIPTION	5
2.1 PRACTICAL BACKGROUND	5
2.1.1 <i>Project owner</i>	5
2.1.2 <i>Previous work</i>	5
2.1.3 <i>Initial requirements specification</i>	6
2.1.4 <i>Initial solution idea</i>	6
2.2 LITERATURE BACKGROUND	7
3 PROJECT DESIGN	8
3.1 POSSIBLE APPROACHES	8
3.1.1 <i>Approach 1</i>	8
3.1.2 <i>Approach 2</i>	8
3.1.3 <i>App development approaches</i>	9
3.1.4 <i>Discussion of chosen approach</i>	9
3.2 SPECIFICATION	11
3.3 SELECTION OF TOOLS AND PROGRAMMING LANGUAGES	11
3.4 PROJECT DEVELOPMENT METHOD	12
3.4.1 <i>Development method</i>	12
3.4.2 <i>Project Plan</i>	12
3.4.3 <i>Risk management</i>	13
3.5 EVALUATION METHODS	14
4 DETAILED DESIGN	16
4.1 HUMAN CENTRED DESIGN	16
4.1.1 <i>Human factors</i>	18
4.2 FRONT-END DESIGN	19
4.2.1 <i>PVT mobile application</i>	19
4.2.2 <i>Admin desktop application</i>	21
4.3 DESIGN TESTING	21
4.3.1 <i>Test plan</i>	21
4.3.2 <i>Test results</i>	23
4.4 BACK-END DESIGN	25
4.4.1 <i>PVT mobile application</i>	25



4.4.2	Admin desktop application.....	28
4.4.3	Web-API and Database design.....	29
5	EVALUATIONS.....	31
5.1	EVALUATION METHODS.....	31
5.2	EVALUATION RESULTS.....	32
6	DISCUSSION.....	35
6.1	APPROACHES.....	35
6.2	CONSTRAINT HANDLING.....	36
6.3	CHOICES AND IMPROVEMENTS.....	36
7	CONCLUSIONS.....	38
8	FURTHER WORK.....	40
8.1	USAGE OF RESULTS.....	40
8.2	WORK AND IMPROVEMENTS.....	40
9	LITERATURE.....	42
10	APPENDIX.....	45
10.1	RISK LIST.....	45
10.2	GANTT DIAGRAM.....	46
10.3	EVALUATION QUESTIONNAIRE.....	47
10.4	USABILITY EVALUATION CONSENT FORM.....	48
10.5	LIST OF ABBREVIATIONS.....	49

1 INTRODUCTION

It is commonly known that fatigue affects vigilance and cognitive abilities. A lack of sleep can in some situations prove dangerous if snap decisions need to be made. This bachelor thesis aims to aid research regarding this issue, mainly to benefit maritime activities.

The application will be used by our project employer, Prof. M. Lützhöft and Western Norway University of Applied Sciences, Dept. of Maritime Studies (DMS), for research purposes. This should be done by utilizing a standardized vigilance test referred to as the psychomotor vigilance test (PVT). More detail on the psychomotor vigilance test (PVT) can be found in Chap. 2.2 Literature Background.

1.1 Goal and motivation

The goal of this bachelor thesis is to create an application for testing fatigue and vigilance. The application also aims to store and organize the result of every test, in a satisfactory manner. In this case that would mean sorted by date and exportable to excel format. For this purpose, the application will use a database that store the data and convert the results into a more readable form for the project employer, such as an excel format. It was desirable that the test be conducted in such a way that it meets the requirements of the already established PVT. The details of the PVT-test and its requirements will be discussed further in Chap. 2.2 Literature background. The overall aim was that the final product should be usable in fatigue research and data collection for the project employer and their research team.

The motivation behind this work was creating an application that can help digitize the PVT, such that the end user can get the test results in a reliable matter. At this present moment this type of application doesn't exist on the mobile platform. Since the PVT test, at the moment of writing, was only available on windows or mac computers, it was rather difficult to use in environments where computers aren't necessarily present. Therefore, the project employers wish for an easy to use application, that can be accessed from their mobile phones. This would improve usefulness, as well as allowing them to conduct tests at several locations with ease. This work was also driven by pursuing quick response time, so to give reliable results. This would deem the results usable for research purposes.

1.2 Context

The idea behind the application came from Prof. Lützhöft from the Dept. of Maritime Studies (DMS). Fatigue is discussed in the maritime domain as a potential risk both in the short term and in the long term. Short-term risks are mainly groundings and collisions attributed to a sleeping watch keeper. Long-term risks include ill health for the ship officers. When working shifts on ships the officers does not always get the rest they need. Additionally, there is an element of unpredictability in shipping, including unpredictable arrival and departure times as well as unpredictable sailing schedules. Unpredictable schedules make it difficult to plan sleep and work, hence the personnel may be too tired to perform their task in a safe manner.

Prof. Lützhöft wants to use the application to test ship officer's vigilance while they are in difference state of fatigue, as if they are at work.

1.3 Limitations

Server Maintenance - One of the limitations of this bachelor project will be maintaining the database server after the project is handed in. We spoke with the IT department at Western Norway University of Applied Sciences, they were not sure if they could provide us with a server and maintenance. Either way they could not implement a solution before the end of our project, creating an uncertain future for the application database. Because of this limitation we had to implement a local storage method as a safety net for the project employer.

Time – A big constraint for this project was the time we were given to develop this application. With the exams finishing in the start of march and the deadline day for the project being 3. June, we had 3 months finishing the bachelor project. 3 months to understand the project, develop the application and with time to spare for writing the report will perhaps limit the result. As the group were new to the Xamarin environment, it can prove to be a learning challenge, therefore impacting the over-all time it takes to produce the cross-platform application of the desired standard.

1.4 Resources

For this project we needed several different resources, working with the different components.

First, we needed a service to synchronize our code, to enable cooperation within the group.

GitHub - This is a platform for developers to work together. It is the most utilizing version controller. GitHub helps you check versions, issues and saving your work in a secure way.

We also needed a developing environment for code compilation, preferably one we're acquainted with already.

Microsoft Visual Studio 2017 - This is an integrated development environment (IDE) from Microsoft. It is used by programmers to develop all kind of computer program, such as mobile applications. It has the live assistants as you write code, no matter what language you use, from C#/VB and C++ to JavaScript and Python. It also comes with an emulator that will help us test our application through the development process.

One of the requirements for this assignment was creating a cross platform application. Therefore, we chose a framework for exactly this.

Xamarin - This is a cross-platform implementations of the Common Language Infrastructure (CLI) and Common Language Specifications (often called Microsoft .NET). The only three things needed for coding cross-platform with Xamarin is Microsoft Visual Studio, C# and .NET, also there are emulator available for Android, iOS and Microsoft. The performance is pretty much the same as native applications, when looking at loading content. Xamarin forms gives you the possibility to give your application the distinct layouts of both Android and IOS, this is important for us when making this application.

Organizing tasks and working with Scrum digitally requires an online accessible platform for the entire group.

Trello – This is a platform for organizing your project in to brackets. Trello will show who's working on what and when, also what is going to be done.

The database used for storing test results remotely needs a management system and language.

MySQL - Is a free, open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). SQL is a popular language for adding, accessing and managing content in a database. It is most recognized for its quick processing, proven reliability, ease and flexibility of use.

1.5 Organization of the report

The report consists of ten different chapters. In order to provide a red line throughout the thesis, the following is a short summary of the contents of each chapter. Additionally, it argues the relations between chapters.

Chapter 1. Defines the goals, motivations and limitations and resources used within the project.

Chapter 2. Describes the project background, the requirements and the initial solution. Also, it determines the project owner(s) and discusses previous work and literature.

Chapter 3. Discusses the different possible project approaches and development methods, as well as defining a project plan. The evaluation methods are also specified here.

Chapter 4. Details the different design choices, both front- and back-end, in addition to discussing the design process of Human Centred Design.

Chapter 5. Presents project evaluations, using the already determined evaluation criteria.

Chapter 6. Presents the results of the evaluation.

Chapter 7. Discusses consequences of the chosen approaches on obtained results.

Chapter 8. Concludes the project. Summarizes the project goals and describe to which extent they were reached.

2 PROJECT DESCRIPTION

In this chapter we'll discuss the background of the project, both practically in Chap. 2.1 and within literature in Chap. 2.2. We will establish the projects owners, discuss previous work that resembles this and the requirements of this project. The initial solution idea is also presented.

2.1 Practical background

2.1.1 Project owner

The project employer, the Western Norway University of Applied Sciences, Department of Maritime Studies, and the developers are shared owners of the project. As the project is to be used for research, the project employer may use the finished product as they please.

The Western Norway University owns the project on behalf of Prof. Lützhöft, whom also came up with the initial idea and would be the one using the application for research for the Department of Maritime Studies.

Prof. Lützhöft is a master mariner, trained at Kalmar Maritime Academy. She has a bachelor's degree in Cognitive science, a master's degree in Computer science and has received a PhD in Human-Machine Interaction. Her field of research includes the effects of new technology and Human Centred Design (Hvl.no, 2019).

2.1.2 Previous work

The planned application builds on the well-established and researched Psychomotor Vigilance Test (PVT). This test has already been tested and used in multiple large-scale studies and is known to be a reliable measurement for vigilance, using lapses in response time (RT) over a given time span (Dorrian, Rogers and Dinges, 2005). The parameters and requirements for this test are quite clear and made for clear guidelines when developing this application. This means that the product will have concise and measurable goals, based on the previous research done regarding the PVT test.

2.1.3 Initial requirements specification

The initial requirements given by the employer was an application that could be used to measure vigilance, utilizing the psychomotor vigilance test. In addition, the test application should store the data, with a timestamp and test identifier, in a satisfactory manner. The data must also be convertible to a readable format, preferably excel.

In addition, there should three different options regarding test duration. The three lengths required were three, seven and ten minutes. The different test length must be stored with the rest of the data, such that one only compares the same length test to each other. This is because the different durations, give different levels of accuracy. The different accuracies regarding test length will be discussed further in Chap 2.2 Literature Background.

Lastly the project employer wished for a cross platform application, so they could run it on any device of their choosing. Therefore, there should be a working version for both android and iOS.

All the specified requirements are regarding the finished product, not the development process. Therefore, the development process can be done with any appropriate tools, as long as all requirements are met.

2.1.4 Initial solution idea

Whilst the PVT is well established, the initial idea for the PVT application came from Prof. Lützhöft, on behalf of the employer Western Norway University. The idea was a response to Prof. Lützhöft research on fatigue and vigilance, especially regarding ship captains, that would be simpler if they could rely on an application to conduct the test needed and collect the necessary data.

The application would be a simple design where you can choose between three different test durations, enter a five-character long identifier and proceed to take the test. The test will at random times allow you to press a button, and log hits, misses and reaction time. When the test is finished, the test will be uploaded to the database. After test completion, the researchers should be able to download and work with the data from the database.

2.2 Literature background

The test design depends on the already established PVT. The PVT is well researched and is known to give reliable results, regarding fatigue. It is among others used by NASA astronauts to monitor the daily effects of fatigue while in space (Nasa.gov, 2019). The PVTs validity is well-documented and has had large participant groups testing its accuracy, with good, all though varying results in different research (Dinges and Basner, 2011). Usually the PVT gives stimuli every 2-10 seconds for a duration of ten minutes. At each stimuli the response time (RT) of the participant is measured. The standard RT limit, before the stimuli is counted as a lapse in attention is 500ms. However, different studies using the PVT has operated with different RT as lapses, or different test lengths, causing differing accuracy in the results. The standard 10-minute test has shown to be highly reliable, using lapses in attention as a key metric, instead of merely using RT as other fatigue tests has done. Hence, the PVT and its performance has been established as arguably the most reliable for measuring vigilance (Dorrian, Rogers and Dinges, 2005).

The standard PVT has a duration of ten minutes, but it has been documented to only lose 22.7% of effect size on average, if you shorten the duration by 70%, which appears to be a sufficing trade-off, considering the time that you save (Basner, Mollicone and Dinges, 2011). Hence, it is apparent that the shorter tests could give satisfying results if the 10-minutes proves impractical in the current setting. A smaller RT being defined as a lapse in these shorter tests could also improve the test accuracy. As a result, the test can be conducted at different durations, but as the accuracy has some variations the different data sets should be kept separate.

During the test the participants RT is tested in random intervals ranging from two to ten seconds. The test records all presses and stores them in the data set as either hits or misses, according to the defined RT of a lapse.

3 PROJECT DESIGN

In this chapter we will discuss the different possible project approaches, and why we selected the ones we did. A project plan is laid out outlining the planned progress, in addition to risk management and evaluation plans.

3.1 Possible approaches

3.1.1 Approach 1

The most important functionality required by the project employer could be achieved with very little difficulty and low risk. It could be done by making an application for both iOS and Android, with identical tests, that allow for local storage. In addition, it had to allow the admin user to export the data to excel on the device itself. The benefits of this approach were that all the project requirements would be met, there would be no need for maintenance, and further functionality could be added if time allowed for it. The downside was that the data collection wasn't automatically centralized, and if the research were conducted using different devices one had to manually merge the data.

3.1.2 Approach 2

The next solution required the application to be developed as in approach 1, apart from how the data would be managed. Instead, this approach would store the data in an external database, and it wouldn't be necessary to access the data locally from the device. This solution would also require a desktop application where the admin could access the data and export it to excel. The benefits of this method would be that it makes the job much easier for the researchers if several devices are in use. The downside is the added risk that comes with more time-consuming tasks, in addition to the need for someone to host and maintain the solution after project completion.

3.1.3 Application development approaches

3.1.3.1 Hybrid application

Building the application in web-code with a mix of HTML5 and JavaScript was one of the possible approaches. This approach would also require a very simple native-application that loads the webpage.

3.1.3.2 Independent native applications

This approach would require two different applications, one for iOS and one for Android. The iOS version would then be written in Swift and the Android version in Java or Kotlin. Note that these programming languages are quite different and learning both would consume notably more time.

3.1.3.3 Cross-platform native application

With the use of Xamarin and the .NET-framework, using C#, we could build a native application which rolls out for both iOS and Android with a little-to-nothing layout difference between the two. This would ensure that the application is near identical for both platforms.

3.1.4 Discussion of chosen approach

3.1.4.1 Overall approach choice

Both the first and the second approach previously discussed has their pros and cons. The choice between them was very much between approach one's simplicity but data storage inconvenience and approach two's higher risk but improved usability. However, the second approach had a couple of risks that were significant to the lasting functionality of the application. As it would have to rely on a database without anyone to maintain it after completion, a flaw could mean the application would be deemed useless, without any local storage solutions. As a result, we chose to start out with approach one, ensuring a stable, local solution, and then if time allowed for it, we'd continue with approach two as added functionality. This combined functionality seemed to combine risk management and a good result in a satisfactory manner.

3.1.4.2 Application development approach choice

Before choosing the desired development approach we had to consider different factors like; learning-curve, design-possibilities, capability of the language, in addition to keeping the test identical for different devices. The last factor is an important aspect of designing applications for research purposes, and this extended to a different factor as well; making sure the responsiveness of the application was satisfactory when high precision were a requirement.

A web-code application was considered because it is quite easy to develop for, and we were already familiar with it to some extent. This would probably prove a worthy method of creating our desired application.

Independent native applications were also under consideration given that we were experienced with developing in Android Studio using Java. Since Western Norway University of Applied Sciences, Department of Maritime Studies, also wanted the application for iOS, we would also have had to research Swift development. This was a language we had next-to-none experience with. We would also likely wind up with building the same application twice which seemed redundant and time-consuming.

The final option, to build a cross platform application, was the option of our choosing. We would build a Xamarin-application, utilizing the .NET-framework. This was a solid solution that allowed us to create only one application which can be used both for Android and iOS. Only the local storage code will differ, and the applications should act identically, with only minor differences between the two appearances. Xamarin is also supported by Visual Studio with a direct plug-in. This is a developing environment we were already familiar with. The main concern we had when choosing this approach was if the Xamarin application would be responsive enough. According to a performance test by AltexSoft (AltexSoft, 2019) Xamarin Forms does not cause any significant delays compared to other native applications. Microsoft also provides guides for creating responsive Xamarin Applications (Docs.microsoft.com, 2019).

3.2 Specification

The reason we chose to start out with approach one for the overall solution, then later add on the functionality of approach two was to ensure that the application would stay usable no matter what. This was mainly a precaution to manage the risk of depending on a server when no maintenance would be performed. Thus, the application will meet all requirements, while maintaining a low risk. It was more important that our project employer got a usable product, than all added functionality. In addition, this approach ensured that we would have a finished product, even if the project turned out to be too time-consuming or we'd run into problems towards the end of the project.

The reason we chose a cross-platform Xamarin application was that it would be a valuable learning experience, creating a cross platform application in a .NET-framework. This with no code having to be written twice in different languages, and no new languages having to be learnt, in comparison to creating two separate native applications. Also, we were all familiar with .NET and C#, and we wished to expand our knowledge within this framework. This would allow us to spend far more time on learning new features within Xamarin, instead of spending time on the language or framework.

3.3 Selection of tools and programming languages

Microsoft Visual Studio 2017 – User-friendly platform for .NET development.

GitHub – Open platform for sharing projects that gives whole team easy accessibility to the latest updates.

Trello – Free platform which easily organises tasks and creates a great overview of the project.

C# - Intuitive programming language developed by Microsoft for their .NET platform and one of the core languages in Visual Studio.

Xamarin – Allows for easy implementation of cross-platform code for iOS, Android and Windows development in Visual Studio using C# code.

3.4 Project development method

3.4.1 Development method

Our development method of choice was Scrum, which is an agile framework for programming. It allows for efficient task management during each project iteration. The core idea was to look at our timeframe and divide the main task into smaller tasks, before dividing these different tasks into different “sprints” through the project. At the end of each sprint at least one piece of the project should be completed. After each sprint the team goes back and evaluates the results of the sprint. This ensures both feedback and evaluation regularly, thus ensuring a better product.

3.4.2 Project Plan

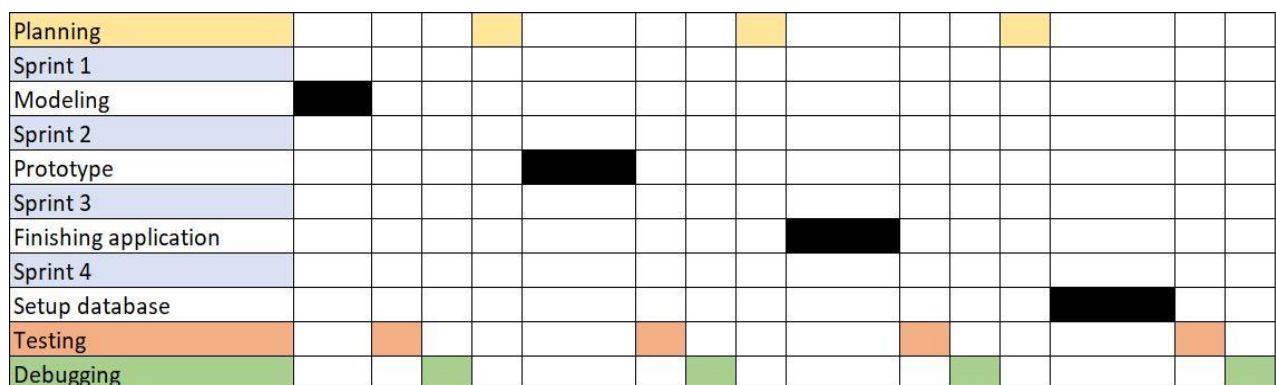


Figure 3-1

Planning

Before each sprint the group planned the task ahead and tried to find the right approach to the upcoming sprint. This included improvements and correcting flaws, from the previous iteration.

Sprint 1

The first sprint was focused on modelling and getting a better understanding of the task at hand. The design was planned, development tools chosen, as well as setting up everything else we needed to get started. All the following sprints were planned out in this phase.

Sprint 2

The second sprint was about building the application shell, with some limited functionality.

This was to verify that the tools, framework and core functionality corresponded with our initial plan.

Sprint 3

The third sprint continued for a longer period and ended with the completion of the base-application without a database setup. After this sprint both the design and functionality were completed.

Sprint 4

The fourth sprint was all about setting up a reliable database and data-transfer for the application. Setting up a method of reviewing and manipulating the test data was also a part of this sprint.

Testing

After every sprint the group executes functional testing on the different sprint's functionality and design.

Debugging

At the end of each iteration we debugged the project. This acted a failsafe with a lot of tests to see if the application and database ran as expected, to test the responsiveness, and fix any potential bugs.

Pre-project report and bachelor thesis

The reports written didn't require a lot of our time compared to the rest of the project, and for that reason wasn't part of the scrum process. The pre-project report aligns with sprint 2, which required some planning to ensure both jobs proceed as planned. This thesis was largely written after project completion.

3.4.3 Risk management

After identifying the risks, we accumulated them in a risk list found in the appendix (Chap. 9.1). This was used to assess each risk individually and plan the necessary precautions. The risk analysis is a technique that is concerned with examining the likelihood that an event will occur, and the significance of the consequences it will bring. After evaluating each risk,

the group had to figure out how to avoid or handle the possible risk event occurring. Based on the risk list the following risks had the highest risk factors.

3.4.3.1 Input-delay from application

In the risk list this case was given 12/25, scoring the liability it could bring the project. Our goal for the applications input-RT was approximately 50 milliseconds at most. To achieve this goal the group took different measures, such as refactoring the code, using threads and changing the event handler for the button from “clicked” to “pressed”.

3.4.3.2 Fail to set-up database

This was also assessed as a 12/25 in the risk list. The reason this case was given such a high-risk factor was because of the groups limited knowledge of working with databases. Throughout this project the group dedicated a lot of time researching other similar projects and applications, to acquire the knowledge of setting up a database properly.

3.4.3.3 Time

The Gantt chart found in the appendix (Chap. 9.2) had to be followed and its deadlines upheld. If time management was to become a problem it could have led to the project falling behind schedule, which could result in a poorly developed solution. As shown in the Gantt chart the group managed to stay on top of nearly every millstone. The group underestimated the time it took to build the prototype, but made up for it by finishing the main application faster than anticipated. This issue was given the highest risk factor of 16/25. To handle this, the project needed high priority by the group members, and a continuous focus on communication was important. This strengthened the group’s ability to identify when extra hours, or a refocus on a specific part of the project, was needed.

3.5 Evaluation methods

Evaluation of our product regarding functionalities and requirements, defined by prof. Lützhöft, was a constant process done through both physical tests and project discussion with the project employer. For thorough testing three approaches were chosen. This was to ensure an evaluation that would incorporate functionality requirements, the project employer’s expectations and the user experience (UX) of the test subjects.

1. Functionality testing, where the developer team makes sure to test that the applications different features are working as intended. This was done at the end of each iteration and would ensure a stable and working application.
2. Design and requirement-discussions in the different stages of the application development process. We presented our current vision or prototype of the application to Prof. Lützhöft since she was our employer, with a long experience in human factor studies. Her opinions then influenced the implementation of each following iteration.
3. The last approach was usability testing for the application by external subjects, with no connection to the development process. Since developers often have a different view of what is intuitive in an application, feedback from the end user can prove quite useful. The hope was that this would provide constructive and direct criticism and let us know of necessary changes. The purpose of this type of evaluation was to make the user interface (UI) as intuitive as possible.

After these different approaches, we aimed to have a complete application solution that was usable for the main stakeholders.

At the end of the project, we would evaluate the finished product by a couple of different metrics. First off, we had the initial requirements as a measurement of how successful the project was. However, this would not provide enough feedback, thus we need additional criteria. Fulfilling all requirements does not necessarily mean the ideal solution is reached, so the finished product would be compared to the project design plans to determine the degree as to which we've reached our goals. Lastly, and most importantly, we were to use the results from the usability testing to determine if the UX was optimal, which was an important part of this project. The usability evaluations were to take place during the last phase of the project.

4 DETAILED DESIGN

This chapter aims to define the solution design in greater detail. Chap. 4.1 explains the Human Centred Design approach used throughout the project. The front-end design is laid out in Chap. 4.2 and later tested for usability in Chap. 4.3. Lastly, the back-end design is presented in Chap. 4.4.

4.1 Human Centred Design

For our design process we've used an approach known as Human Centred Design (HCD). HCD focuses on the end user of the product from beginning to end. It aims to map out disorienting or confusing parts of the design while still in the development process, and then to eliminate these parts before the final product. The result should be an intuitively usable product that gives a good user experience (UX), and that also matches the user's needs. In addition, HCD should consider several issues from human factors, such as universal design, disabilities, previous knowledge, culture and language (Boy, 2017).

The process starts with defining the requirements and planning. This step needs to define an end user profile, as well as a project scope and what exactly needs to be created. Designing the PVT application at this point consisted of planning an early prototype for functional testing, whilst making sure we met all the current design requirements. Since our application were to be used for research, we decided on a simplistic design and interface with as few distractions as possible.

Next up in the development process was the creation. In this step the planned product is developed. This was done in several iterations, along with the rest of the development process. During the first iteration we developed the prototype with basic functionality, to enable us to have a working shell application. This was done using Scrum as a partial process and Xamarin as a development tool.

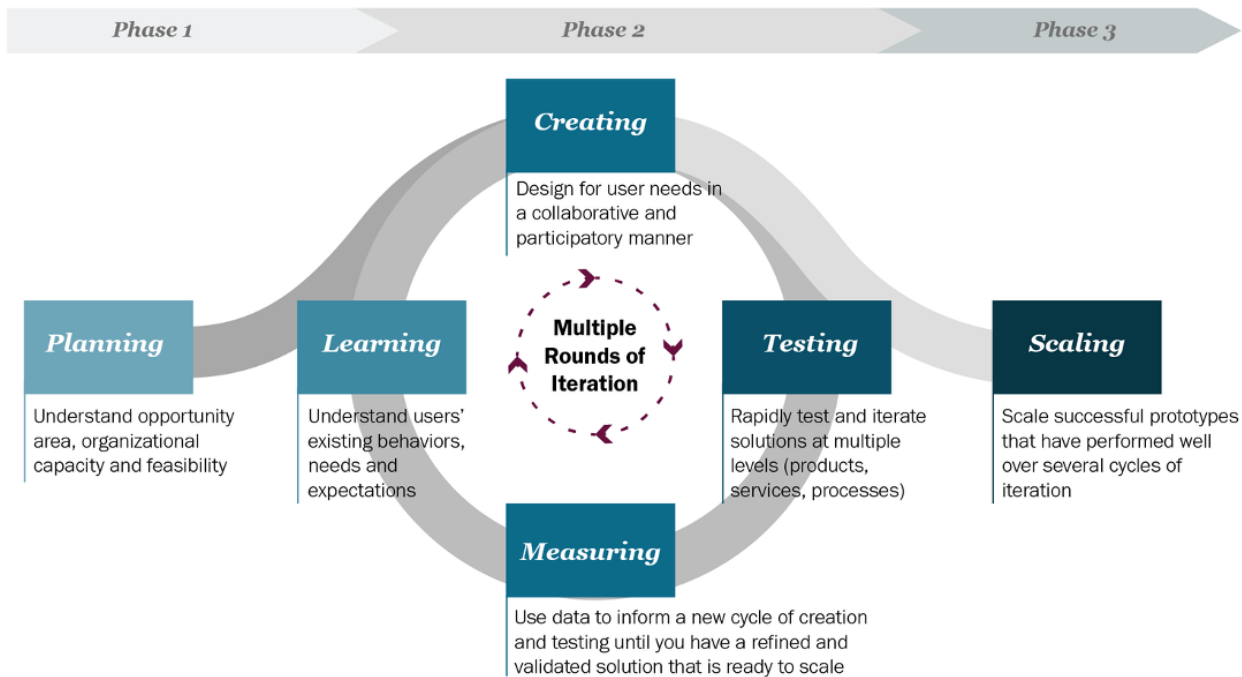


Figure 4-1, HCD process

Testing and measuring what have been created is important to improve on it in the next iteration. One should be able to test functionally, meaning using the application to see how the design works in practice. The results of the testing is then documented and measured, to give a better understanding of what needs fixing. We did this with several demos for our project employer, where we discussed design elements and things that needed improvement. Also, we did a test with several users with no relation to the application, before entering the final iteration.

After testing is done and the data is measured in a satisfactory manner, one must learn from the data and plan the necessary fixes and improvements. In our case this specifically meant changing or removing confusing text, as well as improving several design elements to improve the overall UX.

After a decided upon number of iterations is reached, the created product can be considered finished. For this project we chose to perform a total of four iterations. At this point the product could be deployed, if all functionality has been properly added and tested. This will also take place in iterations alongside the design process, using Scrum. Three of our four iterations went over the design with our employer, and in one of them there were conducted

testing on a slightly larger scale. The result should be a design that is well tested and easy to use.

4.1.1 Human factors

There are several issues influenced from human factors that has been considered while planning and testing the application. These are either human traits we did not want to affect the outcome of the PVT or things we had to keep in mind regarding ease of use for the defined user group. As the project were to be used for research the user group has only very few limitations. The result is the user group being defined as everyone from ages 18 to 70, the maximum retirement age in Norway for nautical professions (Regjeringen, 2014). Everyone in the group also had to be familiar with smart phones. In addition to this, some other exclusions had to be made, mainly people with severe eyesight disabilities, as the reaction test rely on visual stimuli. Also, the user must be able to understand the instructions, which makes reading a requirement.

Some of the most important human factors that impacts usability were the following:

- Preferred hand, as the test should be identical for both right-hand and left-hand users.
- Language.

One of the test requirements was that it should be developed in English. This meant that the user must understand basic English.

- Eyesight.

Whilst users whom suffer from severe eyesight disability cannot use the application, many suffer from reduced eyesight or colour-blindness. These people still had to be able to use the application. Thus, the design choices must reflect this, utilizing sharp contrasts and large text.

- Technological knowledge.

Since the users may come from different backgrounds and age groups it was a fair assumption that their technological competence would differ vastly. Therefore, the application had to sport an easy to use interface, which is understandable for all skill levels. Simplicity would be the deciding factor here.

4.2 Front-End Design

4.2.1 PVT mobile application

The application interface has a simplistic material design. The buttons are flat, on a white background. All text will be black for a sharp contrast. The bar at the top will be blue with white text for Android, while white with blue text on iOS. Thus, the colour palette consists of blue, black and white. The test-page also use the colour red as an indicator for when to press the button, but the design aside from this will be the same. We haven't made any modifications to the font style, which makes it default to the system standard.

On opening the application, you are met by the main page, containing the HVL-logo to indicate that the tests and research is in fact associated with HVL. Furthermore, the page consists of the label “Chose test length”, and three different buttons with the choices of “3 minutes”, “7 minutes” and “10 minutes”. The buttons and label are quite large and in centre of vision to prevent confusion and ensure the user understands the next step. The button design differ slightly between iOS and Android as seen in Figure 4-2 and 4-3. There is also a little gear-icon at the top of the screen. This is for the admin of the application and once pressed a small pop-up appears asking for a password. If entered correctly one is sent to the admin page, which will be discussed further in a bit.

Pressing either of the test lengths takes you to the next page, containing a text-field and an arrow button, with

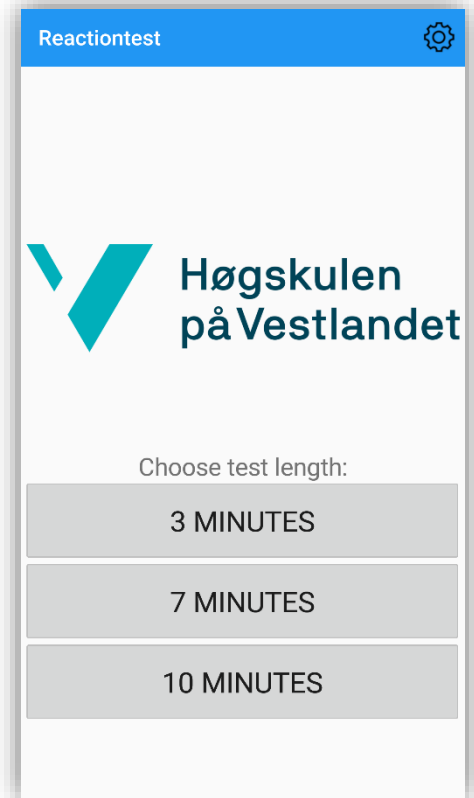


Figure 4-2, Android

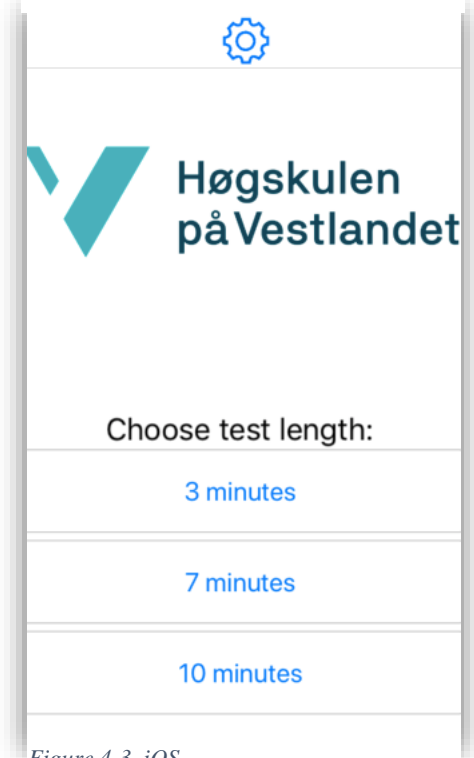


Figure 4-3, iOS

the label “Enter ID here”. After the id has been entered successfully the user is taken to the main part of the application, the test. The test page is blank containing a large button, which is reachable from both sides of the screen and close to the bottom, considering reachability for both hands. The button label reads “Press to start” and above it there are instructions to press when red and wait while grey. The top bar will show the participants username throughout the test, and have a back arrow, in case the user needs to cancel the test. When the button is pressed all text will disappear, as to not distract the test subject and the button will blink red in random intervals. Hits or Misses will be displayed in plain text above the button. When the test is finished a small pop-up will thank the user for their participation and allow them to press finish, which returns to the main page.

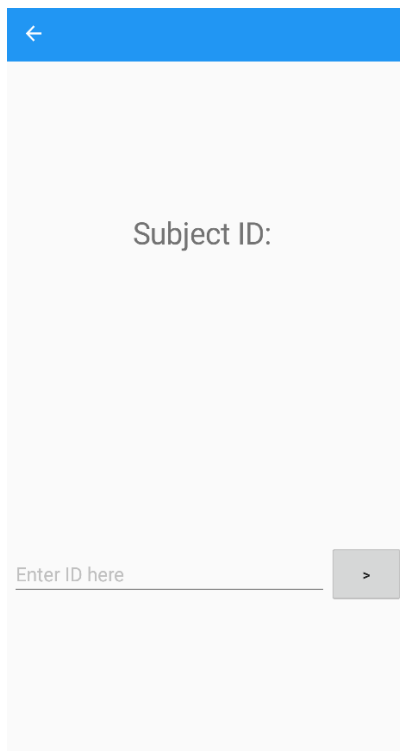


Figure 4-6, User ID page

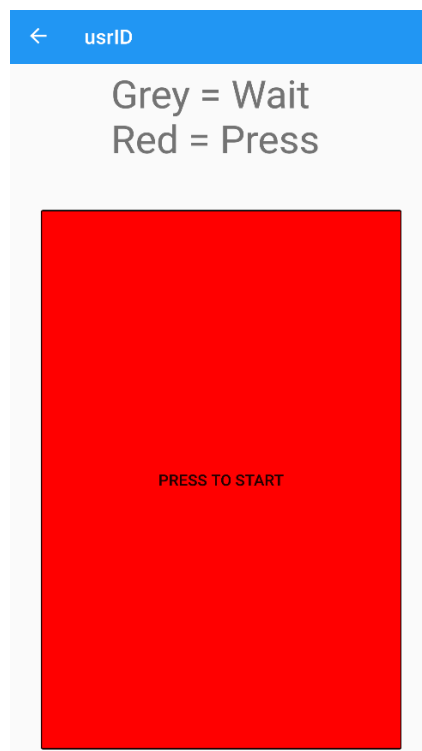


Figure 4-5, test page



Figure 4-4, admin page

The admin page, accessed from the gear icon on the main page, displays two date pickers and a button, in the same style as the rest of the application. The date pickers are labelled “From” and “To”, to allow you to set the time span of your choosing. At the bottom there's button is labelled “Export”. As this page is locked it will only ever be seen by our project employer and those on their team.

4.2.2 Admin desktop application

The desktop application for data management is a simple Windows Presentation Foundation application, only to be used by the researchers. It's one simple page where one can choose start date and end date and/or a user-ID. Furthermore, there's a panel to view all the entries that match the entered parameters and a button to refresh/search for entries. Also, there is a button to export the data as an excel file. The purpose of this is simply to have a way of gathering all the data into one document, from the database.

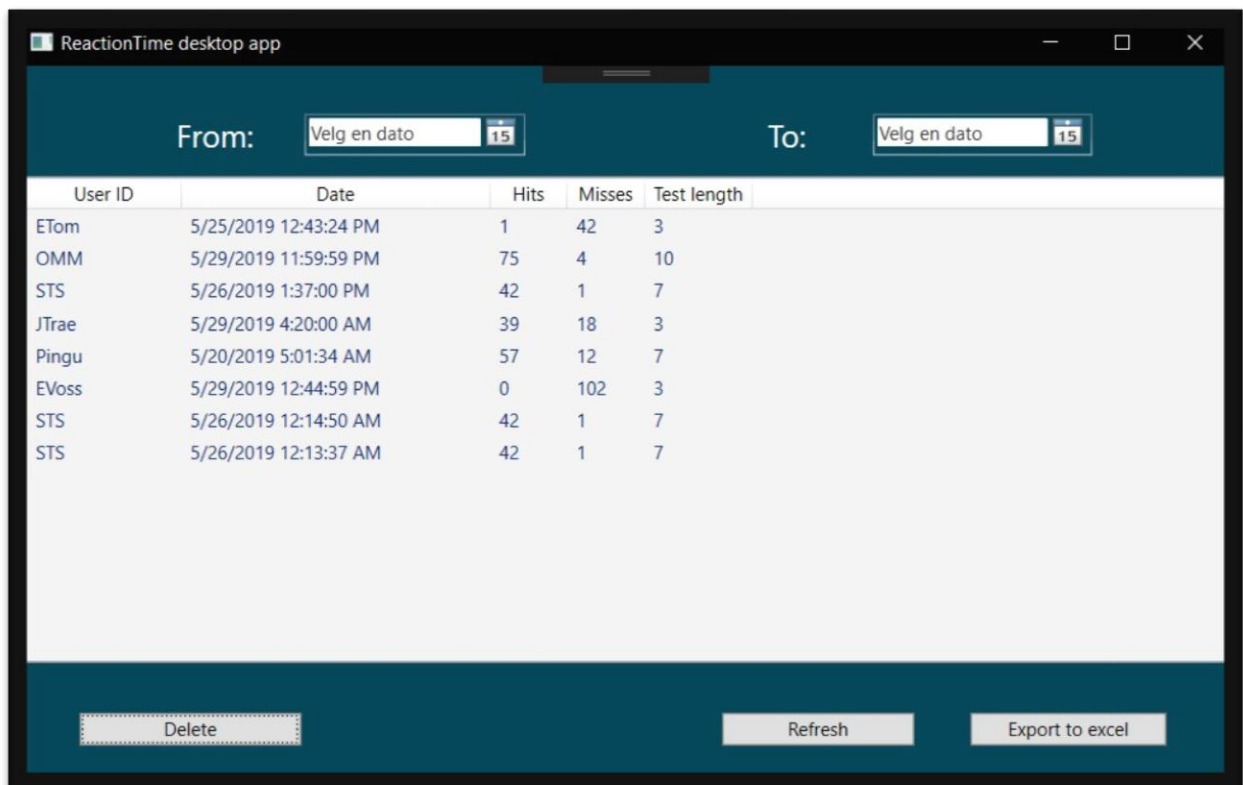


Figure 4-7, desktop application

4.3 Design testing

4.3.1 Test plan

As this is an application where none of the users have much prior experience or can be interviewed as experts on the UX of the application, the best approaches to collect data on the design was either through surveys or usability testing, by end users. We chose usability testing because of the importance of user involvement, and the feedback the user could give that we might never have thought to ask. This were to be done by conducting usability tests

where individuals from within the defined user group perform different tasks, using the application. This would allow us to refine the final product so that it better reflects the users' needs and makes for a much better UX. The testing were to be done in two turns, first one session with a near finished product, then another session with the completed application.

The purpose of the test was to uncover confusing, misleading or bad design aspects of the application.

4.3.1.1 Participants

Usually the more test subjects one can get, the better. But, because of the time constraints of this project we chose to test around eight subjects of ages between 18 and 80. This should be a large enough group to highlight issues that needed fixing in the UX, but not too many to get through efficiently. No subjects should have prior knowledge of the application or an education within IT.

4.3.1.2 Test Method

The test would be performed by handing the subject the phone and telling them to open the application. From there they would be given three different tasks, which they should be able to solve without help within a total of five minutes. If they need help or get stuck, the specific task were marked as failed, but they would be allowed to try the following task, starting from scratch. The only information they were to be given up front was that the application is a reaction test to measure vigilance over an unspecified period of time. The participants was timed on each task and given a score from 0-5, where 0 is a failed test and 5 is perfect execution.

4.3.1.3 Task List

Task 1	Launch 7-minute test
Successful completion	Enters ID, continues and presses start.
Benchmark	User understands navigation. Completes task within 1 minute.
Task 2	Cancel test and return to main page.
Successful completion	Back arrow pressed and screen displays main page.

Benchmark	Completed within 20 seconds.
Task 3	Complete a 3-minute test.
Successful completion	Understands when to press and not. Completes to the end.
Benchmark	3 minutes and 30 seconds completion time.

4.3.2 Test results

The following results are from the 8 participants whom tested the application. The test was performed one on one, as described in Chap 4.3.1 Test plan.

4.3.2.1 Participant 1

Task	Time	Rating (1-5)	Comment
1	10s	5	
2	5s	5	-
3	3m11s	3	Pressed start before reading instructions.

4.3.2.2 Participant 2

Task	Time	Rating (1-5)	Comment
1	20s	4	Confusion regarding not having an ID
2	9s	5	-
3	3m15s	4	Pressed once while grey at first because of impatience.

4.3.2.3 Participant 3

Task	Time	Rating (1-5)	Comment
1	8s	5	-
2	5s	5	-

3	3m7s	3	Didn't read instructions, got to excited by green button, caught on quickly though. (1 "miss" press)
---	------	---	--

4.3.2.4 Participant 4

Task	Time	Rating (1-5)	Comment
1	20s	5	-
2	9s	5	-
3	3m5s	3	Barely read instructions, forgot immediately after the test was started.

4.3.2.5 Participant 5

Task	Time	Rating (1-5)	Comment
1	15s	4	Confusion regarding ID.
2	5s	5	-
3	3m10s	5	Read Instructions.

4.3.2.6 Participant 6

Task	Time	Rating (1-5)	Comment
1	9s	5	-
2	3s	5	-
3	3m6s	5	Read instructions.

4.3.2.7 Participant 7

Task	Time	Rating (1-5)	Comment
1	9s	5	-
2	3s	5	-
3	3m6s	5	Read instructions.

4.3.2.8 Participant 8

Task	Time	Rating (1-5)	Comment
1	9s	5	-
2	3s	5	-
3	3m6s	5	Read instructions.

4.4 Back-End Design

4.4.1 PVT mobile application

As the application was written using Xamarin Forms, the UI is created by XAML files, while the back-end is written in C#. This is mostly event-driven code, which proceeds to update the main UI. For example, on the start page (or the Main menu) a button clicked event sends the user to the next page. The complete navigation flow is shown by figure 4-9.

Most of the code written is for the execution of the PVT test. The test use case is shown by figure 4-10, as a sequence diagram. When the test is started an array of random timestamps is created which determines when the button will turn red and initiate a test instance. A timer is created which runs asynchronously, as to not interfere with anything on the main thread. Whenever the main UI should be updated as a result of background-processes you must invoke the change on the main UI thread, which can be done as follows in figure 4-8.

```

Device.BeginInvokeOnMainThread(() =>
{
    //Code to update a property
});

```

Figure 4-8

It was important to keep the code running on the main UI to a minimum, to keep the application responsive regarding reaction time. This was done by running all timers and background processes on different threads.

When the test is finished the method for saving the results is called. First this saves the results locally as a text file on the respective device, before attempting to send the data to the database. This is done from the `SendToDatabase` class where a json object, containing the test result with all relevant information, is sent using an asynchronous PUT-request to the web-API on the server. This is discussed further in Chap. 4.4.3.

The admin page deals with local storage and exporting to an excel readable format. This is where the main differences between the Android and iOS versions came into play, whilst designing the application. Both devices store the data locally in a similar way, but some issues arise when trying to export the document. The filetype exported is csv which can easily be loaded into excel. This proved to be a more stable way of solving the export than converting the data to excel format from within the application, as that required several custom libraries, and differed vastly from iOS to Android.

4.4.1.1 Android

The Android OS allows the user to access folders and files from the device itself. This means we can save the csv-file directly to a folder on the phone, that the user has access to. This is done using a separate class for the Android application that does exactly this.

4.4.1.2 iOS

On iOS the user can't access files stored directly on the device. Therefore, our iOS application has a slightly different solution. Instead of storing the csv-file locally, a share function has been implemented where the user can save the file to their personal iCloud account. This is done using a separate share class within the iOS application.

4.4.1.3 Flow chart

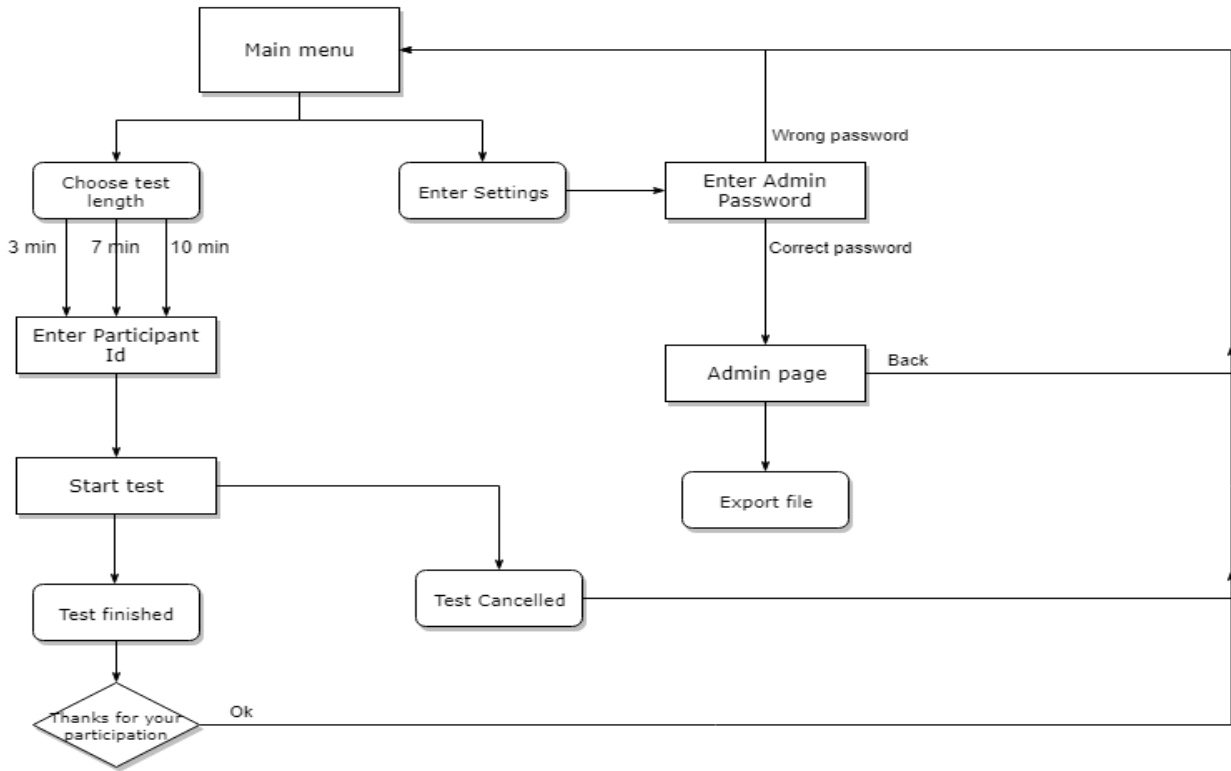


Figure 4-9, flow chart

4.4.1.4 Sequence diagram, test

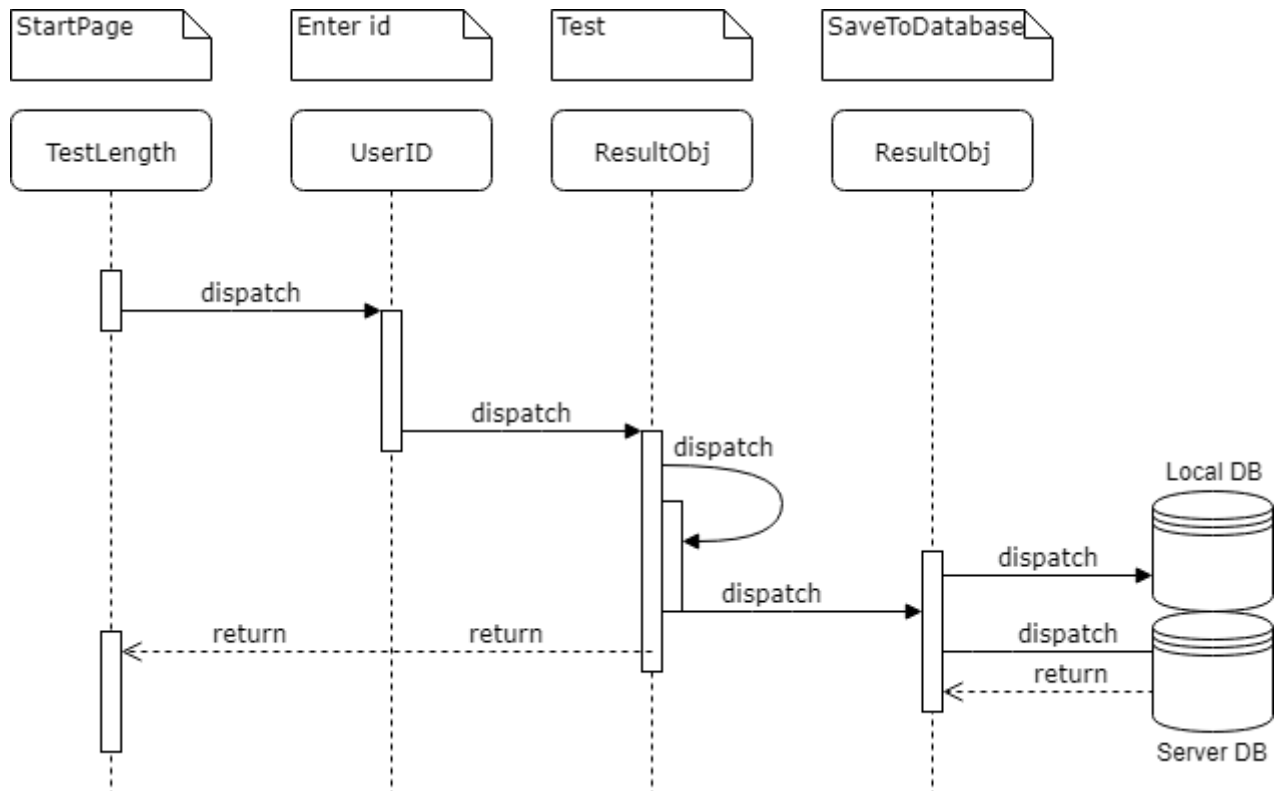


Figure 4-10, sequence diagram

4.4.2 Admin desktop application

The admin desktop application was designed for the admin to both delete entries and export wanted entries from the database to excel-format. The application is made in Windows Presentation Foundation (WPF) which makes it easy to develop a simple application for Windows. The test-data is extracted from the database through requests to the Web-API, residing on the server.

The application consists of 7 key elements; a list-view for displaying the different tests, 2 date-pickers for filtering tests between 2 wanted dates, one text-field for filtering by user-ID and 3 different buttons.

The first button is for deleting a test-instance. The code behind it fetches the selected item from the list-view, gets the ID of the item and sends a delete-request to the Web-API accompanied by the ID. The second button is for refreshing the list-view. This includes filtering by user-ID and filtering between selected dates (if filled out). In addition, the refresh button sends a new get-request to the Web-API, to fetch any new tests that might have occurred since the start of the program. The last button is labelled “Export” and simply exports the data into a new excel file with one simple click. To achieve this, we utilized Microsoft Office’s Interop Excel (Microsoft, n. d) which made this possible with only a few lines of code.

4.4.3 Web-API and Database design

When the test is finished the application sends an HTTPS request to the Web-API including all the data needed for the database in JSON format. In the API we have implemented the Data-class under Models, as a blueprint for how the database will look. From there we have auto-generated the controller and data context. In the Controller folder the data-controller class is located. This class is where the HTTPS request is handled. The POST method is called by the request from the application. This method is validating the JSON formatted data by matching the variable type of this data, and the variable type data initialized from the Data-class in the API. If all required variables are filled, the request goes through and the API and sends the data to the database.

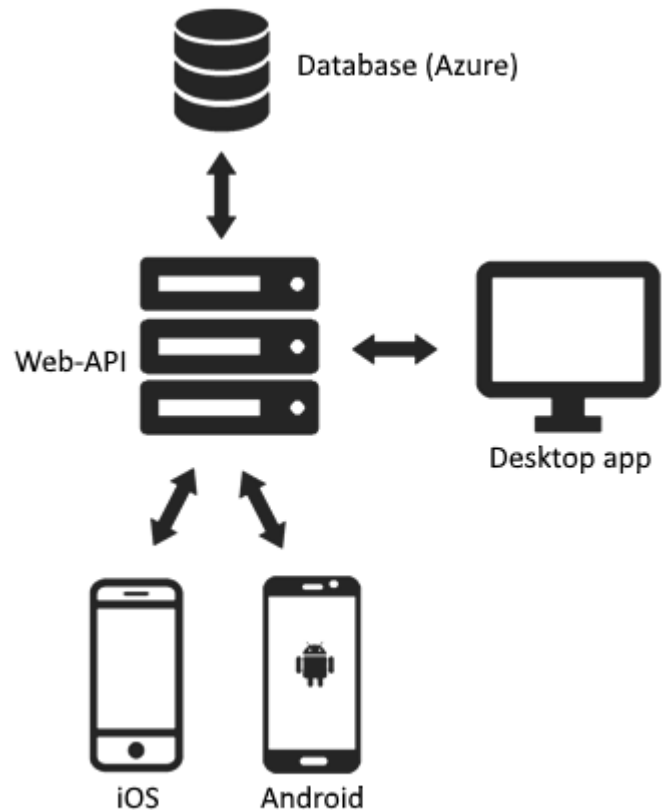


Figure 4-11, connection overview

At the start of this project we thought we would be able to use the Western Norway University of Applied Sciences' (HVL) server, but this was not as easy as we thought. But we were given a temporarily virtual machine (VM) we could use. The now working solution utilizes this VM as a server, with both the Web-API and database stored locally. If the project employer would like to use the application after one year having everything on the VM could be a problem. Therefore we hope that the WNU can move everything from the VM to their own server.

The database design utilized was a simple set-up, containing only one table as shown in figure 4-12. After each test is finished all the data is sent from the application to the API and

stored in one row in the database. The data being stored in each separate column is testId, User-ID, date, hit, miss, test-Length and 100 reaction times measured in milliseconds. The group was uncertain that using 100 columns to store each individual test click was the most efficient way to organize the data, but the project employer found that to be the preferred way for them to manage and view the results.

The date object is a DateTime variable that includes the date and time in milliseconds for when the test is finished. Our first solution was to use this as the primary key for the database, but this was since scrapped because of the risk of 2 phones completing the test at the same time. This would cause one of the results being thrown away. Instead we are now using a simple auto-incremented ID, this also makes it easier for when we want to delete entries in the database, with just having to refer the ID instead of the exact date and time.

Results
+ int: testId
+ string: userId
+ DateTime: date
+ int: testLength
+ int: hits
+ int: misses
+ double: test1
+ double: test2
+ double: test3
...
+ double: test100

Figure 4-12, database

5 EVALUATIONS

In the following chapter the different evaluation methods used throughout the project are presented and discussed (Chap. 5.1). It then goes on to summarize the evaluation results in Chap. 5.2.

5.1 Evaluation methods

Evaluating the project using the best possible methods is crucial to obtain a good understanding of the final product, as well as the decisions that lead there. In addition, the evaluation method of choice should have some impact on the development process itself. If the evaluation were used only after the fact then no benefits will be given the project, and the only use for it will be whilst drawing a conclusion. Therefore, a good evaluation process was utilized during the project development, as well as at the end to judge the result.

Our evaluation method consists of three different parts.

A criteria-based evaluation was the first step towards judging the project. The criteria used to judge the project were first and foremost the initial requirements and if all have been fulfilled. In addition to this, the different components in the project design and their specified functionality were required to complete all criteria. Altogether, this forms a checklist for the project:

1. PVT application for both iOS and Android.
2. 3-, 7-, and 10-minute test.
3. Ability to export to excel format.
4. Completion of Approach one (local export).
5. Completion of Approach two (database running with desktop application).

Upon completion of the following points, the product could be deemed finished. This checklist would also be used as goals or milestones during the project. However, this approach only evaluates how much of the project were finished, not the quality of it, thus more evaluations are necessary.

Secondly, the usability tests performed near the end of the project served as a useful source of evaluation. Four of these tests were done using the almost finished product, while four were done with the final result. This made the method useful in both for improving the application, and for evaluation of the result. The purpose of this kind of testing is mainly quality assurance, specifically regarding the UX. If the UX don't meet the end users' expectations this may cause unnecessary frustrations or errors. In a neutral research environment it is vital that the application is easy to use and understand. The issues different subjects encountered gave insights into how the application is used and their feedback was a valuable data source while evaluating the project. The details of the usability test are listed previously under Chap. 4.3 Design Testing.

Lastly, an evaluation will be done by Prof. Lützhöft, the project employer. As she had the initial idea for the product and is the one using the application for her research, it was important that her expectations were met, and she was satisfied with the product. To ensure this there was held regular meetings where the UI was presented, and she's gave relevant feedback and criticism. This was to ensure that the development-process never deviated to far from her vision. In addition to feedback on the UI she's been involved in all major decisions regarding test development, so that it matches the research plans as closely as possible. All evaluation meetings have been documented along the way, so that all decisions are retraceable and reasoned. These constant evaluations are a good tool for ensuring that the project moves in the right direction, but upon completion of the project it was also important that the project employer evaluated the result. This was done by preparing a questionnaire for the employer to fill out, documenting her thoughts on the project. This gave them a last opportunity to judge the result, in comparison to their expectations. This is perhaps the most crucial part of the evaluation as the project employers will be the ones using the product.

5.2 Evaluation results

The criteria-based evaluation resulted in all five of the previously specified points being achieved, although with a few minor configurations. The export of the document was done using csv format instead of excel, in the local storage solution. The reasoning behind this is

that it was much easier while developing, in addition to being compatible with Excel. Thus, this is not considered as a large drawback during evaluation.

The results of the usability tests gave some interesting insights that helped improve the application. The relevant results can be seen in Chap. 4.3.2 Test Results, Participant 1-4. Even though the tests were done on a near-finished product, some flaws were uncovered. Several of the subjects *pressed to start* the test prematurely, and as a result didn't get to read the instructions. Some confusion ensued, where many did a test click which was characterized as a miss. This was especially inconvenient as it disrupts the data collected. One of the subjects also reasoned that the green button made her click instantly, because of her association between green and *go* or *ready*. The changes that ensued was changing the colour to red, having more negative associations, and making the instruction text endure for a couple of seconds after the test is started. There was also some confusion regarding the subject ID, which none of the subjects had been given, but all of them eventually typed their name and completed the task. In fact, this seems like the appropriate reaction from a test subject to the ID question, as the researchers prefer choosing the IDs beforehand. Apart from these minor changes the UX was good, where all subjects navigated through the application and the test intuitively.

After the issues uncovered from the usability test were fixed, a new round of testing ensued. Here the participants 5 to 8 were given the exact same tasks as in the first round of testing. We saw great improvements during this round of usability tests. The previous observed blunder of starting the test instantly, before reading instructions, didn't occur once this time around. This was partially thanks to the start button now being red, making the subject pause for a second, before launching the actual test. In addition, the instructions not disappearing at once, but rather showing until the first stimuli were presented, seemed to eliminate any remaining confusion after the test had started. As seen from the test-scores of the specified participants, there were significant improvements from the first iteration of tests. During the first round the participants had an average score of 13/15 total, which suggests there were some smaller issues in the UX. This increased to the a 14.74/15 average in the second trial, suggesting near perfect execution and an improved UX from the earlier version. The

confusion around not having an ID persisted, but no measures were taken to prevent this as the ID should be provided to them by the researchers.

The regular meetings with the project employer resulted in an application where they knew to a large extent what to expect, such that there would be no big surprises or misunderstandings when the project came to an end. After the final product was handed over to the project employer, the results of the questionnaire gave the project a final evaluation, judging the different aspects of the product. The questionnaire that was used can be found in the appendix (Chap. 9.3). The project employer was asked to rank the different parts of the project on a scale from 1 to 5, where 1 is the worst and 5 the best outcome. There were also two open ended questions where the employer could criticize and suggest improvements, or highlight parts they were particularly happy with. Each one of the ranking questions were given a 4/5, meaning near complete satisfaction with the application, but still room for some improvements. The section for criticism was originally left blank, but in a later email an improvement was suggested. When the keyboard appears to enter the user ID, it overlays the text field, obscuring the view. Fixing this has been added to the Chap. 7.3 Further work. The section for positive feedback only mentioned “student dedication”, which is more of a complement to the group, than the product itself. In summary the employer’s evaluation gave an average score of 4/5 and suggested one minor improvement to the product.

6 DISCUSSION

The following chapter aims to discuss the different choices of the project, and their consequences. First, we discuss the different approaches in Chap. 6.1, then the limitations and their impacts on the project in Chap. 6.2. Lastly, in Chap. 6.3 we discuss choices that could have improved the product.

6.1 Approaches

Initially we chose to complete approach one, local export, before moving on to approach two, database storage, when creating the application. This increased the workload and led to a shorter than desired timeframe for getting the database up and running. Although this made the project a bit larger, we felt it was worth it having the local storage solution as a back-up. Furthermore, this ensured that the product would still be usable even if future problems with the database are to appear. The result was added application functionality, paired with less reliance on a single storage solution. The additional feature also makes the application usable in an offline environment. This makes for a more robust application, which is beneficial for our project employer.

We've developed the application in the .NET framework, using Xamarin. At first this seemed like an ideal solution, as we already had some familiarity with .NET, but during the project it became apparent that we had a lot to learn about Xamarin. We encountered several time-consuming problems, which would have been far simpler to solve in a framework we had more experience with. One example was simply making the UI update the way we wanted, which ended up taking an unnecessarily large amount of time. Another issue was the different local storage solutions we had to implement for iOS and Android, within the same application. These were issues that would have been easily resolved if we chose to make a hybrid application, with web-code. However, we learned far more by using Xamarin and don't regret this decision, despite the increased workload. In the end we reached all the desired functionality, with identical tests on both iOS and Android.

The Human Centred design approach had a positive impact on the project, as the end user always was the focal point. All the way from early plans, up to final product, the end user

always drove the decision-making process. The result was a UX both we and the project employer are very satisfied with. This approach to developing an application also paired well with our use of Scrum and iterative development.

6.2 Constraint handling

The time constraint served was a major concern throughout the project. It is demanding balancing good result and rapid deadlines. The main result of this was testing and debugging getting a lower priority than desired. Although the application and its design has been tested functionally, very few tests have been done code wise. This was something we wish we had more time for. Another problem regarding time were how we underestimated the database setup part of the project, thus setting aside far too little time for it while planning. Ideally, we'd start working on the database earlier and as a result have more time in the end, perhaps also for testing and debugging.

The project limitation, server/database maintenance, has been a motivation in the project to have more than one stable storage solution working. Despite this increasing the workload slightly, this limitation drove us to developing two separate solutions, to ensure a good result.

6.3 Choices and improvements

The way the database structure was organized, was very inefficient and could see some improvements. Whilst it worked well on a small scale, for larger projects it would be an unsuitable solution. The use of only one table and no unique participant identifier is an unconventional solution, but as it worked on this scale and was a wish from the project employer, it was an acceptable solution. If the research were to be conducted on a larger scale however, it would be beneficial to split the database table into four, one for each unique participant, and one for each of the three different test lengths.

One of the initial requirements for the application was that the export format would be excel. However, while developing in Xamarin this led to a lot of issues and turned out to be a lot more time-consuming than first presumed. In response we made the choice to switch from excel-format to the simpler csv-format. This still technically fulfilled the requirement as csv-

files can be loaded into excel, but it would be ideal to export directly to excel. If the project's time frame was longer, adding excel-support would be a reasonable improvement for the product. But as this choice saved us a lot of time and acted as a fair replacement, it seemed like an acceptable trade of.

7 CONCLUSIONS

The following chapter will summarize the initial goals, and then go on to draw a conclusion based on Chap. 5.2 Evaluation results. Furthermore, we will discuss how this thesis can benefit other projects and future work to be done on this one.

The goal of this project were to create a mobile application to measure vigilance and fatigue, for both iOS and Android. This were to be accomplished using the PVT as a template. The test results would be recorded and the data had to be exportable to an excel readable format. During the development process Human Centred Design principles would be considered, aiming for a good UX and an intuitive UI.

As seen in Chap. 5.2 Evaluation Results, all the desired functionality of the application was reached. This means a functional PVT application joined by a desktop application and server-database was successfully created, connected by a Web-API. The only possible drawback is that for the local storage solution csv-files was used instead of excel-files, however this was not the case on the desktop application. As previously discussed in Chap. 5.2 this was deemed an acceptable trade of. Thus, it can be concluded that all planned functionality was reached in a satisfactory manner.

In the design evaluation we relied on feedback from the usability tests and from Prof. Lützhöft. This was to ensure that the users understood the interface and had a good experience, in addition to meeting Prof. Lützhöft expectations.

The usability test on the near finished product uncovered a few flaws, which were corrected before the next round of tests. As discussed in Chap. 5.2 Evaluation Results, the second round of testing came with improvements and a near perfect result. The users navigated the application with ease and performed the test flawlessly. Hence, the usability test was an essential tool to polish the UI and test as much as possible. Considering the results from the second round of usability testing, we can conclude that the UX was very good, and the PVT was intuitive to use.

The final evaluation from Prof. Lützhöft regarding the application design was done using the questionnaire found in the appendix (Chap. 9.2). The results of this evaluation were

discussed in Chap. 5.2 Evaluation results. To recap, the average score given was 4/5, meaning near complete satisfaction with the results. There was also one small suggested improvement. Judging from these results the product seems to have lived up to the employers expectations in all aspects. Not getting a perfect score suggests the product could have been a little better, but there is usually always room for some improvement. Nevertheless, given the projects time constraints, this seemed like a perfectly sufficient evaluation score. The suggested fix didn't appear to be a large issue either, as this was something easily resolved and could be updated in the future with little effort. Hence, the project employers evaluation also deemed the completed project satisfactory.

8 Further Work

8.1 Usage of Results

The obtained results can in some cases be useful for other projects our purposes. There are at least two imagined cases where one can learn from this thesis.

When setting up a PVT this thesis can provide all necessary specifications to successfully conduct the test. All the relevant information needed to perform a PVT can be found in Chap. 2.2 Literature background. The exact parameters normally used for the PVT can sometimes appear scattered while doing research, if one does not have any background in fatigue and vigilance studies. Hence, the summary provided could prove quite helpful if one were to develop a PVT in the future.

Preparing a project using HCD or similar approaches, Chap. 4.3 Design Testing could prove as a useful case study. This way of conducting usability testing can prove useful, if planned correctly and interpreted well. Other development teams could use a similar testing template, to hopefully achieve good results.

8.2 Work and improvements

For now, the Web-API runs on an Azure server, along with the database, provided by Western Norway University, Faculty of Engineering and Science. This is a paid service, and despite working properly the project will either have to be moved to a server of the project owners choosing or the project owner must agree to pay for the service. The further work would be moving the database and API to a different server, with support for C# and MySQL.

The way the application was constructed it first saves the data locally, then tries to it send it to the database. However, if the device does not have an internet connection it will only be stored locally. It would be preferred that the data is sent to the database whenever a stable connection is detected, but this has yet to be implemented. If further work were to be done with the application, this improvement would have a high priority.

The current test sees about a 40ms latency, which is not bad compared to our goal, but still not perfect. Future improvements could be desirable to increase the test accuracy. This could

be done by writing unsafe code (C++), which is a more efficient language. Though, this requires some advanced thread handling, which would have been too time consuming for the projects timeline.

A final fix suggested by the project employer, regarded the keyboard overshadowing the text input on the user ID page. This is something that could be fixed by making the text field move dynamically when the keyboard appears or simply moving it further up on the screen as a default. Either way, this would not require a lot of effort and should be a simple fix to a simple problem.

9 LITERATURE

Agogino, A., Levine, D. and Lesniewski, M. (2015). Design Thinking in Development Engineering. [online] Best.berkeley.edu. Available at: http://best.berkeley.edu/wp-content/uploads/2015/07/DE_Mudd_v16_AMA.pdf [Accessed 25 Apr. 2019].

AltexSoft. (2019). Performance Comparison: Xamarin.Forms, Xamarin.iOS, Xamarin.Android vs Android and iOS Native Applications. [online] Available at: <https://www.altexsoft.com/blog/engineering/performance-comparison-xamarin-forms-xamarin-ios-xamarin-android-vs-android-and-ios-native-applications/> [Accessed 21 May 2019].

Basner, M., Mollicone, D. and Dinges, D. (2011). Validity and sensitivity of a brief psychomotor vigilance test (PVT-B) to total and partial sleep deprivation. [online] Available at: <https://www.sciencedirect.com/science/article/pii/S0094576511002256#bib12> [Accessed 1 Apr. 2019].

Basner, M. and Dinges, D. (2011). Maximizing Sensitivity of the Psychomotor Vigilance Test (PVT) to Sleep Loss. [online] PubMed Central (PMC). Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3079937/> [Accessed 27 May 2019].

Boy, G. (2017). The Handbook of Human-Machine Interaction. Milton: CRC Press, pp.3-16.

Codecademy (n.d.) what is a Relationship Database Management System? [online] Available at: <https://www.codecademy.com/articles/what-is-rdbms-sql> [Accessed 31 May 2019].

Dalberg.com. (2019). What is Human-Centered Design? | Dalberg. [online] Available at: <https://www.dalberg.com/what-human-centered-design> [Accessed 30 May 2019].

Dinges, D., Dorrian, J. and Rogers, N. (2005). Psychomotor Vigilance Performance: Neurocognitive Assay Sensitive to Sleep Loss. [online] Med.upenn.edu. Available at: <https://www.med.upenn.edu/uep/assets/user-content/documents/Dorrianetal.PVTchapterinKushida2005.pdf> [Accessed 27 May 2019].

Git (n. d) Git [Online] available at <<https://visualstudio.microsoft.com/xamarin/>> [accessed at 31. may 2019].

Hvl.no. (2019). Margareta Holtensdotter Lützhöft - Høgskulen på Vestlandet. [online] Available at: <https://www.hvl.no/person/?user=6018126> [Accessed 27 May 2019].

Khitrov, M., Kumar, K., Reifman, J., Liu, J. and Ramakrishnan, S. (2018). PC-PVT 2.0: An updated platform for psychomotor vigilance task testing, analysis, prediction, and visualization. [online] Available at: <https://www.sciencedirect.com/science/article/pii/S0165027018301109> [Accessed 1 Apr. 2019].

Microsoft.com. (2019). .NET | Microsoft University. [online] Available at: <https://www.microsoft.com/nb-no/about/university/dotnet.aspx> [Accessed 30 May 2019].

Microsoft (2019). Writing Responsive Applications - Xamarin. [online] Available at: <https://docs.microsoft.com/en-us/xamarin/android/app-fundamentals/writing-responsive-apps> [Accessed 21 May 2019].

Microsoft (n. d.) Microsoft.Office.Interop.Excel Namespace [online] Available at <<https://docs.microsoft.com/en-us/dotnet/api/microsoft.office.interop.excel?view=excel-pia>>[Accessed 27 May 2019].

Microsoft (n. d) Xamarin App Development with Visual Studio [online] available at: <<https://visualstudio.microsoft.com/xamarin/>> [accessed at 31. may 2019].

Microsoft(25/01/2018) Windows Presentation Foundation [online] Available at: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/> [Accessed 31 May 2019].

Nasa (n. d.). NASA - Psychomotor Vigilance Self Test on the International Space Station. [online] Available at: https://www.nasa.gov/mission_pages/station/research/experiments/982.html [Accessed 1 Apr. 2019].

Regjeringen.no. (2014). Pensjonsordning for arbeidstakere til sjøs. [online] Available at: <https://www.regjeringen.no/contentassets/5f2a740ecab04b5da1f52ea94e2f903c/no/pdfs/nu201420140017000dddpdfs.pdf> [Accessed 30 May 2019].

SearchSoftwareQuality. (2019). What is Scrum? - Definition from WhatIs.com. [online] Available at: <https://searchsoftwarequality.techtarget.com/definition/Scrum> [Accessed 30 May 2019].

SQL (n. d) What is MySQL Tutorial [Online] available at <<https://www.siteground.com/tutorials/php-mysql/mysql/>> [accessed at 31. may 2019].

Techopedia(n.d) What is an Integrated Development Environment (IDE)? [online] Available at: <https://www.techopedia.com/definition/26860/integrated-development-environment-ide> [Accessed 31 May 2019].

Techopedia(n.d) What is an Common Language Infrastructure (CLI)? [online] Available at: <https://www.techopedia.com/definition/24365/common-language-infrastructure-cli> [Accessed 31 May 2019].

Trello (n. d) What is Trello? -Help [Internet] available at <<https://help.trello.com/article/708-what-is-trello>> [accessed at 31. may 2019].

Visual Studio. (2019). Xamarin App Development with Visual Studio | Visual Studio. [online] Available at: <https://visualstudio.microsoft.com/xamarin/> [Accessed 1 Apr. 2019].

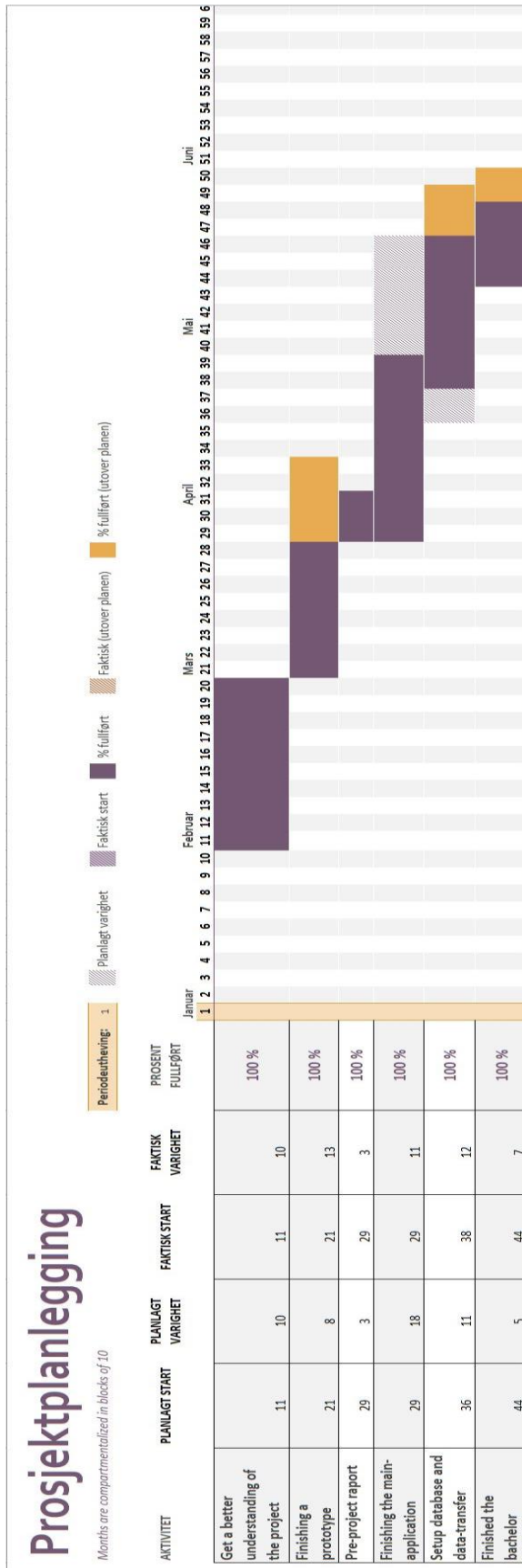
Visual Studio. (2019). Editing Code in Visual Studio IDE | Visual Studio - Visual Studio. [online] Available at: <https://visualstudio.microsoft.com/vs/features/ide/> [Accessed 1 Apr. 2019].

10 APPENDIX

10.1 Risk list

Risk	P	C	R	Mitigation
Not accomplishing iterative tasks within the time limit	4	2	8	Set up a proper plan for the different iterations and tasks within the iterations. And if it should occur the different iterations could overlap.
Input-delay from application	3	4	1 2	Try out different approaches and if that doesn't work, try a different programming language.
Failed to set-up database	4	3	1 2	Reading into other projects and other applications source code.
Problems communicating within the group	1	2	2	Planning fixed meeting times.
Difficult to communicate with the Internal advisor	2	3	6	Be aware of the problem and try to make each other understood.
Difficult to communicate with employer	3	3	9	Frequently planed Skype-meetings.
Problems with the tests. Not achieving valid feedback.	2	3	6	Ensure well-defined tests with specified tasks and data collection metrics, to give measurable results.
Time	4	4	1 6	Define milestones and try to stay within our deadlines.
Difficult to test for iOS	2	4	8	Since none of us own a computer which runs the macOS or an iPhone, we will have to borrow equipment from other students to test the product.

10.2 GANTT diagram



10.3 Evaluation Questionnaire

1. To which degree has the application met your requirements?

Not at all - 1 2 3 4 5 - Completely

2. How satisfied are you with the application design?

Not at all - 1 2 3 4 5 - Completely

3. How satisfied are you with the application functionality?

Not at all - 1 2 3 4 5 - Completely

4. How satisfied are you with the application over all?

Not at all - 1 2 3 4 5 - Completely

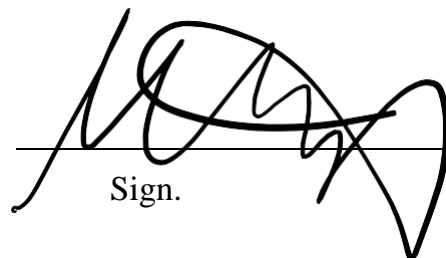
5. How satisfied are you with the solution over all?

Not at all - 1 2 3 4 5 - Completely

6. Do you have any further suggestions or criticism?

7. Is there anything you're particularly happy with?

Student del d'v cartoon


Sign.

10.4 Usability Evaluation Consent Form

Usability Evaluation Consent Form

In this usability evaluation:

- You will be asked to perform three tasks using the PVT-app.
- You will be asked to comment on the user experience of each task.

Participation in this usability study is voluntary. All information will remain strictly confidential. The descriptions and findings may be used to help improve and evaluate the PVT-app. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

If you have any questions after today, please contact Eirik F. Tømmervik at eirik.tommervik@gmail.com. Please sign below to indicate that you have read and you understand the information on this form and that any questions you might have about the session have been answered.

I agree to participate in the study conducted by the Eirik Tømmervik, Sindre Steinsvik and Oskar Midbøe.

I understand that participation in this usability study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.

Date: 27.05.19

Please print your name:



Please sign your name:



Thank you!

We appreciate your participation.

10.5 List of Abbreviations

CLI - Common Language Infrastructure, is a Microsoft specification for running high-level language program applications in different computer systems without changing the application code (Techopedia, n. d).

DMS - Dept. of Maritime Studies.

HCD – Human Centred Design.

HVL - Western Norway University of Applied Sciences.

IDE - Integrated Development Environment, is an application eases application development. A tool for developing (Techopedia, n. d).

PVT - Psychomotor Vigilance Test, is among one of the most popular behaviour alertness tests.

RDBMS - Relational Database Management System, is a program that allows you to create, update, and administer a relational database. Most relational database management systems use the SQL language to access the database (Codeacademy, n.d).

RT - Response time.

SQL - Structured Query Language is a programming language used to communicate with data stored in a relational database management system. SQL syntax is similar to the English language, which makes it relatively easy to write, read, and interpret (Codeacademy, n.d).

UI – User interface is the visual representation of the application that the user interacts with.

UX – User experience.

VM - Virtual machine.

WPF - Windows Presentation Foundation provides developers with a unified programming model for building line-of-business desktop applications on Windows (Microsoft, 25/01/2018).