Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Jørgensen, Trym
Sunde, Andreas Rojahn
Søbakken, Anders
Torland, Eskil

EEG for ALS Bachelor Thesis

Bachelor's project in Engineering in Computer Science
Supervisor: Saleh Abdel-Afou Alaliyat
Co-supervisor: Kai Erik Hoff

May 2021

NTNU
Norwegian University of
Science and Technology

Jørgensen, Trym
Sunde, Andreas Rojahn
Søbakken, Anders
Torland, Eskil

EEG for ALS Bachelor Thesis

Bachelor's project in Engineering in Computer Science
Supervisor: Saleh Abdel-Afou Alaliyat
Co-supervisor: Kai Erik Hoff
May 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Kunnskap for en bedre verden

NTNU

Kunnskap for en bedre verden

DEPARTMENT OF ICT AND NATURAL SCIENCES

IE303612 - BACHELOROPPGAVE

EEG for ALS Bachelor Thesis

Authors:

ANDERS SØBAKKEN
ANDREAS ROJAHN SUNDE
ESKIL TORLAND
TRYM JØRGENSEN

Supervisors:
SALEH ABDEL-AFOU ALALIYAT
KAI ERIK HOFF

Resource Person:
ROBIN T. BYE

19th May 2021

Group Declaration

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Publication Agreement

Supervisors: Saleh Abdel-Afou Alaliyat and Kai Erik Hoff

Credits: 20

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven §2).
Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.
Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☒ **ja**	☐ **nei**
Er oppgaven båndlagt (konfidensiell)? (Båndleggingsavtale må fylles ut)	☐ **ja**	☒ **nei**
- Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	☐ **ja**	☐ **nei**
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	☐ **ja**	☒ **nei**

Dato: 19.05.2021

Summary

The purpose of this thesis was to improve the quality of life, better communication, and improve independence for people affected by ALS or other physical disabilities. A mobile application was created which is controlled by EEG-Brainwear from Emotiv along with a desktop application which connects the mobile app and headset together.

The project was carried out using the development methodology Scrum and it was decided to use Flutter for creating the mobile app, and Electron with React to create the desktop app.

The result of this thesis includes a mobile application released on App Store and Google Play Store. As well as a desktop application, a website and a Flutter library. The main features we implemented to reach our project goals for the app were the smart solution Philips hue and text-to-speech. Our goal for the desktop app was to make connecting to the mobile app with the EEG-Brainwear as easy and intuitive as possible.

Overall, the bachelor group is satisfied with the outcome of this thesis. We think the applications can be used to better communication between people affected by physical disabilities and the people close to them, as well as improving their degree of independence.

Sammendrag

Målet med denne bacheloren var å forbedre livskvaliteten, bedre kommunikasjonen og uavhengigheten til mennesker rammet av ALS eller andre fysiske funksjonshemninger. For å oppnå disse målene ble det utviklet en mobilapplikasjon som er kontrollert ved bruk av EEG-utstyret. I tillegg ble en skrivebordsapplikasjon skrevet for å koble mobilapplikasjonen og utstyret sammen.

Vi bestemte oss for å bruke en utviklingsmetode kalt Scrum i prosjektet, og for å bruke Flutter til å utvikle mobilapplikasjonen, og Electron sammen med React for å utvikle skrivebordsapplikasjonen.

Resultatet av denne bacheloren inkluderer en mobilapplikasjon som ble utgitt på App Store og Google Play. I tillegg til dette, en skrivebordsapplikasjon, en nettside og et Flutter bibliotek. Hovedfunksjonen vi implementerte for å nå prosjektmålene med mobilapplikasjonen var smart løsningen Philips Hue og text-to-speech. Målet med skrivebordsapplikasjonen var å gjøre tilkoblingen til mobilapplikasjonen med EEG-utstyret så lett og intuitiv som mulig.

Alt i alt, så er bachelorgruppen fornøyd med resultatet av denne bacheloren. Vi tror at applikasjonene kan bli brukt til å bedre kommunikasjonen mellom personer som er rammet med fysiske hemmelser eller ALS og personene som står nær dem, og bedre deres grad av selvstendighet.

Acronyms

ALS Amyotrophic Lateral Sclerosis. iii, 1, 2, 4, 7, 69, 70, 76–78, 85

API Application Programming Interface. 1, 6, 9, 21, 25, 39, 43, 49, 58, 61–65, 71, 75, 80, 85

AVK Accessible Virtual Keyboard. 1, 77

BCI Brain-Computer Interface. 1, 5, 22

CPS Cyber-Physical Systems Lab at the Department of ICT and Natural Sciences. 1

CSS Cascading Style Sheets. 1, 28–31, 71

EEG Electroencephalography. iii, 1, 2, 5, 6, 14, 21–23, 35, 49, 77–79

GUI Graphical User Interface. 1, 6, 17, 72

HTML Hyper Text Markup Language. 1, 28–31, 71

IP Internet Protocol. 1, 6, 13, 36, 38, 40, 41, 47, 51, 65, 66

JSON JavaScript Object Notation. 1, 25, 28, 65

REST Representational State Transfer. 1, 65

SDK Software Development Kit. 1, 30, 58, 70, 71, 79

Glossary

Application Programming Interface An Application Programming Interface is an interface used to let two applications communicate. 1

Domain Name A domain name is an unique internet address. An example of a domain name is google.com. 1, 66

Frontotemporal Dementia Frontotemporal dementia is a type of dementia that affects language, behaviour, and personality. 1, 4

Hyper Text Hypertext is text with references, also called hyperlinks, to other text that the reader can immediately access.. 1

Markup Language Markup Language is a computer language. Markup Language uses tags and contains standard words, rather than typical programming syntax. 1

Message Event The message event is fired when data is received through a WebSocket. 1, 45

NullPointerException A NullPointerException is a RuntimeException. The exception is thrown when an application attempts to use an object reference that has a null value. 1, 28, 72

Props Props is a special keyword in React. Props stand for ”properties” and is used to pass data from one component to another. 1, 9

REST API A Representational State Transfer Application Programming Interface is an API that follows the different constraints of the REST principle. 1, 62

Steady State Visually Evoked Potential Steady State Visually Evoked Potential is the change in EEG readings caused by a visual stimulation.. 1, 69

useState A useState is a Hook that allows the developer to have state variables in a functional component. When the state is changed the components re-renders. 1, 72

Waterfall Model The waterfall model is a linear project management approach. The waterfall model relies on teams following a sequence of steps and not moving forward until the previous phase is completed. 1, 17

Contents

List of Figures

Listings

1 Introduction

This report will review the possibility for a person with *Amyotrophic Lateral Sclerosis*(ALS) disease or impaired functioning, to communicate and control certain smart solutions with the use of *Electroencephalography* (EEG) equipment. The purpose of this project was to create an app that could improve the quality of life for those suffering from ALS. We will be using the EEG-brainwear from Emotiv in this project.

1.1 Background

In recent years the *electroencephalography* (EEG) technology has become cheaper and more available. Companies such as Emotiv have created commercial EEG devices aimed at the general public. This enables EEG reading at home. Previously EEG was used in hospitals to examine different conditions in the brain, such as brain disorders, especially epilepsy or another seizure, brain tumor, stroke and more [23].

This project was issued by *the Cyber-Physical Systems Lab at the Department of ICT and Natural Sciences* (CPS) at NTNU in Ålesund. Students at NTNU in Ålesund have previously written several bachelor theses [48] [68] [2] about EEG-brainwear and arrived at different concepts on how to apply the brainwear to different systems. There has been developed several programs aimed at helping people with ALS. One of our goals were to implement some of the previous solutions into our application and improve on them. The group aims to improve the independence and quality of life of persons suffering from ALS. Therefore, the group also wanted to research different possibilities to implement smart solutions into the application.

1.2 Objectives

The project issue is to make the best possible use of the EEG-brainwear to create an application for a person with the ALS-disease to improve their communication with family, friends and health personnel, as well as improve their independence with the help from smart-solutions.

The primary objectives are:

- Develop a mobile application that runs on the iOS and Android operating system.
- Enable the user to control the mobile application by reading their brain activity.
- Develop a desktop application that runs on the Windows and macOS platform.
- Develop a desktop application that connects the Emotiv headset to the mobile application.
- Provide opportunities for the application users to communicate with others.
- Provide text-to-speech functionality.
- Provide a keyboard designed for typing with brain activity.
- Integrate some sort of smart solution into the mobile application.

1.3 Limitations

For this bachelor thesis, a requirement has been set to use headsets from Emotiv. This also implies using the Emotiv App and EmotivBCI software.

1.4 Report Structure

1.4.1 Chapter 1 - Introduction

Introduction to the bachelor project, including background, objectives, goals, and limitations of the project.

1.4.2 Chapter 2 - Theoretical Basis

A theoretical overview of ALS, EEG, existing technology, architecture of the system, design patterns and principles, and the network protocols used in this thesis.

1.4.3 Chapter 3 - Materials and Methods

Materials and Methods describes everything needed to recreate the project at a later time. This chapter includes technical aspects such as frameworks, programming languages, and development methodology, as well as how the project was organized and planned.

1.4.4 Chapter 4 - Results

An objective description of the results found in this bachelor project. This includes both the mobile and desktop applications, a Flutter library, and a website.

1.4.5 Chapter 5 - Discussion

A subjective assessment of Materials and Methods, and Results. It discusses limitations and deviations compared to the original plan and objectives. Technical results, choice of technology, further work and experiences from the project are discussed as well.

1.4.6 Chapter 6 - Conclusion

A conclusion of the project based on the previous chapters, including a summary of whether the goals were reached or not.

2 Theoretical Basis

This chapter describes the theoretical basis and old research that the bachelor is built upon. It also includes design principles, concepts, and theory used to achieve the best result possible.

2.1 ALS

ALS is a neurodegenerative disease that destroys nerve cells in the spinal cord and brain. The ALS disease goes under the umbrella term *motor neuron disease*. The gradual degeneration of these cells causes a loss of motor control. According to the American hospital, Hospital for Special Surgery, it does not affect a persons ability to think clearly. On the other hand, Helsenorge says that a large number of people affected by the disease may experience Frontotemporal Dementia. However, both sources agree that ALS does not affect any sensory functions. Frontotemporal Dementia affects language, behaviour, and personality [66] [29].

In the early stages of the disease this may be some dysfunction in an arm or a leg, or making it harder to talk and swallow. As the disease progresses, most patients will experience all these symptoms [66].

ALS can be split into two categories of disease. Upper motor neuron disease, which affects the nerves from the brain all the way down to the spinal cord. The other category, called lower motor neuron disease, affects the nerves that goes from the spinal cord out to the rest of the body[66] [29].

Every person afflicted with ALS have a different amount of upper and lower neurons that die over time. This results in symptoms that varies a lot from patient to patient in the early stages of the disease. As the disease progresses the patient usually falls into one of two categories of ALS: bulbar-onset ALS, or spinal-onset ALS. These are still the same disease, with the difference being where the disease starts developing the most. Bulbar-onset ALS affects the throat and mouth, while spinal-onset ALS affects a persons limbs, back, stomach and thorax. A person with ALS will experience all these symptoms as the disease progresses. ALS is 100 percent fatal and current medicine will only comfort the

patient and slow down the development of the disease, therefore development and usage of new technology is important [66] [29].

2.2 EEG - Electroencephalography

Electroencephalography is a way of measuring and reading the electrical activity of the brain. These readings are possible through pairs of electrodes that is attached to the scalp. The pairs then sends the difference in voltage between them to an electroencephalograph. Using EEG readings it is possible to show the activity of the brain and see different results as the person using it performs different tasks. The readings can be used to detect different disorders such as epilepsy and other seizures. However emotions, thoughts, and other complex functions of the brain can not be related to EEG patterns, as only electrical activity from the surface of the brain is recorded in electroencephalography. Electroencephalography was first studied in 1929 by a scientist named Hans Berger [23] [52].

Through a *Brain-computer interface* (BCI) it is possible to send EEG readings into a machine learning model that translates the readings into specific commands. These commands can then be used in technologies such as robotic arms to move it around, to steer a wheelchair, or control a mobile application using brain signals instead of muscles [54].

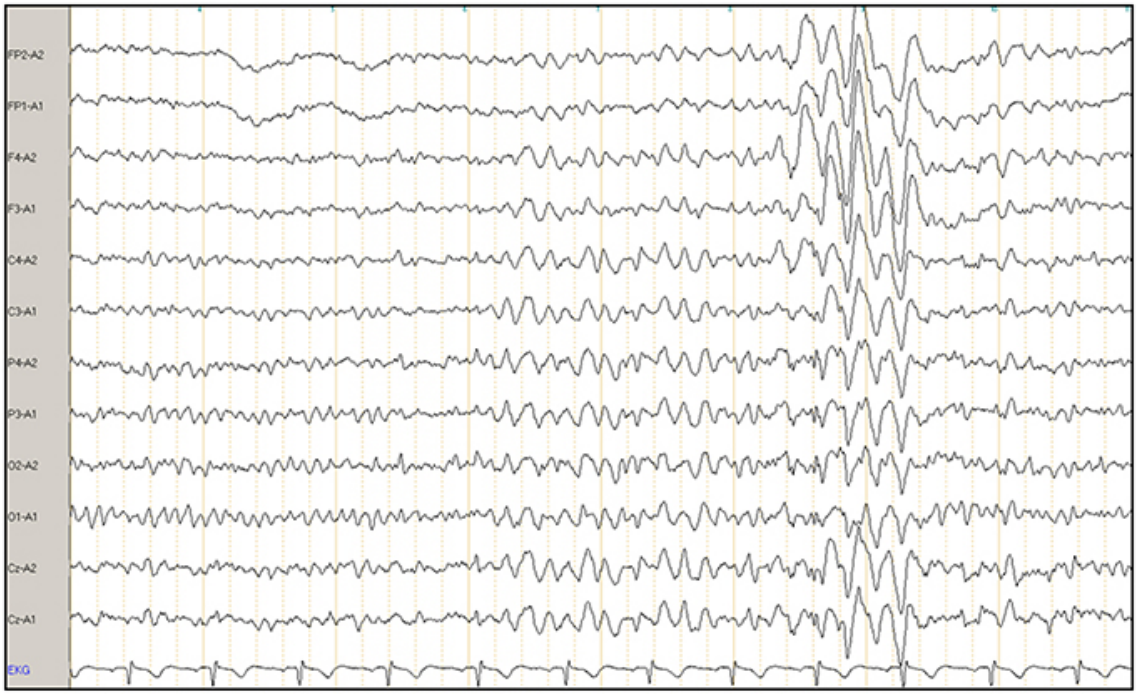


Figure 1: An example of an electroencephalogram reading. Source: [62]

2.3 Existing Technology

When developing a product it is important to look at already existing technology, either to find inspiration or to make sure the product is not already developed.

2.3.1 Emotiv

Emotiv is a privately owned bioinformatics company that focuses on EEG and using EEG to train and better understand the brain. Emotiv has a machine learning model that can translate the EEG signals into commands that can be used for steering when implemented. Emotiv have implemented the possibility to use 15 mental commands where four of those can be active simultaneously [16].

2.3.2 EEG Accessible Virtual Keyboard - 2017 Bachelor

The bachelor project is built upon the bachelor project from 2017, which resulted in an android application and some research about EEG, and using it in mobile applications. The 2017 bachelor group implemented a keyboard that was optimized for use of the Emotiv Epoc Headset. Different keyboard layouts was a key feature of the bachelor, and they used algorithms to optimize layouts. They also conducted user tests to compare the theoretically most efficient keyboard to the most efficient keyboard in practice [2].

2.3.3 Previous Desktop Application and Drivers

Two students at NTNU Ålesund were tasked with updating the drivers and create a new GUI for the 2017 bachelor 2.3.2. Emotiv updated their API and the old drivers were outdated. The students had 10 weeks to develop the application. They wrote the drivers in the programming language C# and used the Windows Forms GUI framework to create the user interface.

The result was a Windows application. The application was split into two programs. The first application was the user interface and the other was an application that ran in the terminal window. The user interface offered the user an option to add the IP address of

the phone and to select their training profile. The terminal was used as a way to give the user feedback about the processes. If an error occurred it was printed in the terminal[46].

2.3.4 SimpliHere

SimpliHere is a mobile application that focuses on making care-taking of patients with ALS easier. SimpliHere have implemented a push to talk function for navigating the app. SimpliHere also has other features, such as a forum to talk to other people diagnosed with ALS, and links to buy equipment[55].

2.3.5 Next Mind

Next mind is a company that produces neuroheadsets. These headsets translates brain activity into commands that can execute simple tasks. Next mind uses Unity to create an applications that can communicate with the headset. The Unity application uses different buttons that are tagged with a graphical overlay. The overlay is trained with the Next Mind machine learning model. The learning model can pick up which button the user is focusing on and activate it. [44]

2.3.6 Neuralink

Neuralink is a company founded in 2016 by Elon Musk. The company created a neural implant interface called The Link. The Link consists of 1500 electrodes that read information from brain activity. In 2020, they started testing the devices on monkeys and managed to read simple inputs such as Up, Down, Left and Right. Neuralink will focus on implementing this device on humans in the coming years. They will use it as an opportunity to test how well the solution is working and improve it further. If the tests results are positive, they will implement control of smart devices, such as smart TVs and mobile phones through The Link [20].

2.4 System Architecture

The software architecture of a system is a depiction of said system. It is a high level description of the system and the components and describes how they will behave and interact. The software architecture of the products is decided in the early development stage.[21]

It is important to keep the scope of the architecture in mind when deciding on the software architecture of the system. The goal of software architecture is to set the field for the developers, as well as a way to communicate how the system works for stakeholders. Changes to the system can be costly once a software is implemented. Good software architecture removes the need to make costly changes. Bad system architecture can come in the form of buggy implementation or flawed design. Identifying these bugs and flaws as early as possible is more cost efficient than fixing errors over the course of the entire development [21][61].

2.5 Design Patterns

Design patterns is a general and reusable solution to a problem that is commonly occurring. Design patterns are not a finished design that can be transferred into code. Applying design patterns to your application or project can save time. Design patterns uses common known ideas that have been improved on over the years. Another benefit from using design patterns is that it can improve the readability of the code. Implementing well known design patterns can help other developers recognize the purpose of objects and classes. In this section the group will introduce the design principles that was used to achieve the bachelor thesis result [57].

2.5.1 Singleton Pattern

The singleton pattern is a well known design pattern in software design. A singleton restricts the class initiation to one single instance. When only one object is necessary across the entirety of the system, Singletons can be useful [57].

2.5.2 Observer Pattern

The observer design pattern is one of the most popular design patterns in software design. The observer design pattern belongs to the *practical design patterns* and is a way to define a consistent one-to-one dependency between one or more objects. The observer design patterns works with two actors; the subject and the observing objects. The subject is the actor of which the status is observed. It also notifies the observers when the state changes. The observing objects, also called *observers*, is the object that get notified when the state of the subject changes.

The observer pattern enables the object to quickly and simply relay all changes made of the subjects state to the observers. The observing object does not need any information regarding the observers, since the interaction is completed regardless of the observers interface. The observing object can receive updates automatically instead of asking for the status of the subject at a regular interval. It results in a system where there are not any unsuccessful requests, since the subject has not changed [32].

2.5.3 Facade Pattern

The facade design pattern is a design pattern that can be used as a class to provide a simplified Application Programming Interface(API) for a complex module. A facade can be used as a buffer for a collection of classes or a library and provide a simpler use for the same functionality. The facade pattern can also be used to avoid tightly coupled clients and subsystems. [26]

2.5.4 React Container Pattern

The container pattern is commonly used to separate components that fetch data from presentational components. The container component fetches and passes the data as *Props* and renders the sub-components. The pattern is useful since it makes it easier to share and reuse the components [67].

2.6 Don Norman's Design Principles

There are many different principles when it comes to designing a software system, but Don Norman's may be the most important to remember and abide by. Norman's main idea is that devices, computers and interfaces should be functional, easy to use, and intuitive. Don Norman introduced the six principles, Visibility, Feedback, Affordance, Mapping, Constrains, and Consistency. Below are descriptions of each of the design principles [19].

2.6.1 Visibility

The first design principle is *visibility*. Visibility tells user what their options are and how to access them. If it is easy to find a button or functionality in the software, then it is more likely that the user will use it [19].

2.6.2 Feedback

The second principle is *feedback*. Just as Newton's third law of motion, it is expected that every action has a reaction. When the user interacts with something it is expected that the user receives some kind of indication that something happened. The indication can be a sound, a moving dial or a spinning wheel to show the user that their action caused a reaction [19].

2.6.3 Affordance

The third principle is *affordance*. The connection between the looks and usage of a component is called Affordance. It is important that the user knows how to use something as soon as they see it. A good example for a product with high affordance is a mug. It is easy for the user to figure out intuitively how to use it [19].

2.6.4 Mapping

The fourth principle is *mapping*. Mapping is the relationship between what something looks like and how it is used. If the software has a good design then the controls to something will closely resemble what they affect. An example of good mapping is the vertical scroll bar [19].

2.6.5 Constraints

The fifth principle is *constraints*. Constraints are the limits of an interaction or an interface. A good example for constraints is a grayed out button. If a button is grayed out, it indicates to the user that the button is disabled [19].

2.6.6 Consistency

The sixth principle is *consistency*. Consistency is the principle that the same action should cause the same reaction every time. If a website has a back button, the user expects the button to navigate to the previous page. If the button navigates to the previous page sometimes and to a random page other times it would be hard to navigate the website [19].

2.7 Programming

A programmer should aim to write maintainable code which is easy to test and redesign. These concepts are important to follow to create well written code and a good end result. The concepts used in this project are mentioned below.

2.7.1 Coupling

Coupling refers to how different modules depend on each other. Modules should be as independent as possible from other modules. If the code has high coupling, then the module know way too much of other modules. The affect of high coupling is that if one module needs modifications, it would influence the other modules as well. The dependency on

other modules creates more work for the developer. Whereas, if you aim for low coupling you are able to easily make internal changes to a module, without worrying about the impact it would make on other modules. In addition, low coupling also makes it easier to design, write and test the code. The reason being that the modules are not interdependent of each other. Another benefit of low coupling is that the developer get reusable and compose-able modules [42].

2.7.2 Cohesion

Cohesion refers to how the elements of a module belong together. Code that is easy to maintain usually has high cohesion. To achieve high cohesion it is important that the elements within the module are directly related to the functionality that the module is meant to provide. If the code has high cohesion it is easier to write, design, and test the code. If the code has low cohesion it means the code which makes up the functionality is spread all over the code-base. In addition, it would be hard to discover what code is related to your module [42].

2.7.3 Component Testing

Component testing is testing each component of a software individually and separated from other components. The main goal of a component test is to ensure that an element of the system does what it is supposed to do. A benefit of using component testing is if all the underlying elements of a system does what they are supposed to do, there will be a smaller chance that the complete system has flaws [27].

2.7.4 Unit Testing

Unit testing is testing small parts, usually methods, of a software individually. The tests are constructed to confirm if they behave as expected. There are a lot of benefits to unit testing. Testing improves the code quality by catching bugs early, which then saves time for the developer. Good unit tests will tell the developer if expected behaviour breaks. Thus, the developer does not have to be afraid of breaking the behaviour without knowing and leaving it in the code [69].

2.7.5 Usability Testing

Usability testing is testing the design and layout of an application or software. The goal of the usability test is to record the test participant as it performs the different tasks in the test. Through these results it is possible to identify bugs, find improvements, and get feedback about the layout and design [36].

2.8 Network Protocols

A protocol is a set of rules which is used to communicate with other devices through the internet. Protocols are standardized through a formal agreement through international forums like W3C and ISO. The standardization make it possible for two devices to understand each other. The network protocols used in this bachelor project are described in this section [11].

2.8.1 WebSockets

WebSockets is a bidirectional protocol. It means that communication goes both ways, from the server to the client and vice versa. WebSockets is also a stateful protocol, which means the connection between the server and client will be kept alive until it is terminated by either one of the parties. If one of the parties closes the connection, the connection will terminate for both parts. WebSockets are useful for application who need real-time data. In WebSockets, data is continuously transmitted into an connection that is already open. It makes the WebSockets faster and improves the application performance [5].

2.8.2 Internet Protocol

The Internet Protocol or IP is a protocol for routing and addressing packets of data. The Internet Protocol ensures that the data arrives at the correct destination. An address inside of the Internet Protocol is called an IP address. When a device is connected to the internet, it is assigned a unique identifier in the form of an IP address [11].

3 Materials and Methods

Materials and Methods includes everything needed to recreate the project, from start to finish. It describes the planning phase, what development methodology was chosen, EEG headsets used, and applications from Emotiv needed to use the headsets. This chapter also describes the software architecture, programming languages, frameworks, software development kits, and libraries used. How the project was managed is included as well.

3.1 Planning

A thorough plan of the project was needed to complete a project of this size in the short time frame of the bachelor thesis. The project started with brainstorming ideas about the methodology, programming languages and frameworks. The brainstorming process resulted in a preliminary report and charts about the project structure.

3.1.1 Preliminary Report

A preliminary report is a document created in the early phase of a project used to get a clear view of the project and how it will be completed. The preliminary report usually includes a description of the problems and goals, and how the goals will be reached through steps. Our report also includes rules and norms about how much, how often, and how the project members should work. The report also states how internal conflicts should be solved and a risk analysis was created for the project. A risk analysis of potential hazards and challenges helps ensure the team is confident in case of equipment failure or unforeseen events. For example firmware failure and Covid-19 restrictions. Risk analyses also make sure minimal time is wasted. The group has split the project into smaller tasks and added a time frame for each task. This was the inspiration for the groups first Gantt chart [53].

3.1.2 Charts

Gantt Chart

Gantt chart is a useful way to plan, schedule and track tasks in larger projects. Tasks are usually on the left side and a calendar are on the right. The calendar shows when each task should be started and finished, as well as the progress of the tasks. Figure 2 shows a part of our Gantt diagram and how we constructed it. The group tried to split the project into tasks that lasted between three and seven days. However, some tasks needed a shorter time frame and some needed longer. The group kept updating the Gantt charts throughout the project as new features needed to be implemented or new problems arose. The Gantt chart helped the group focus on the parts of the project we were supposed to be working on.

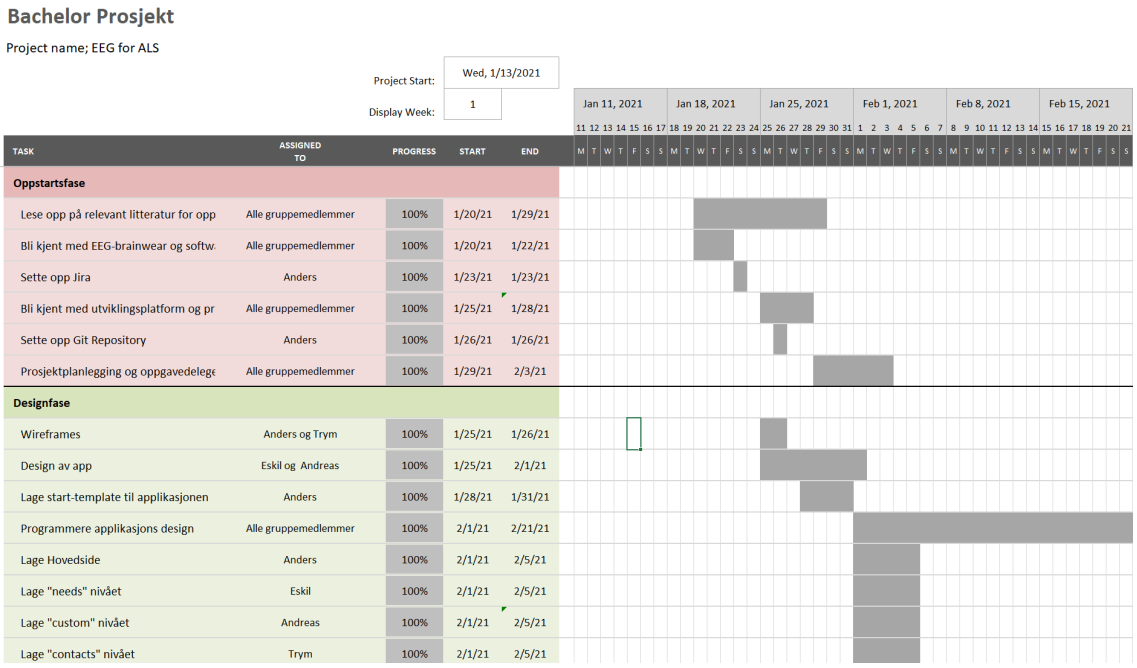


Figure 2: Example of our Gantt chart

Wireframes

A wireframe can be compared to a blueprint. The wireframe shows the page structure, layout and user flow in an application. Wireframes are used to give applications a good structure before any code is written. A wireframe layout saves a lot of time compared to fixing the layout after it is already programmed. It is important to note that wireframes focus on the structure, and therefore lack color and details. They are not to be taken as

the final design of an application. Figure 3 is an example of one of our wireframes. The depicted wireframe is for the keyboard page in the application.

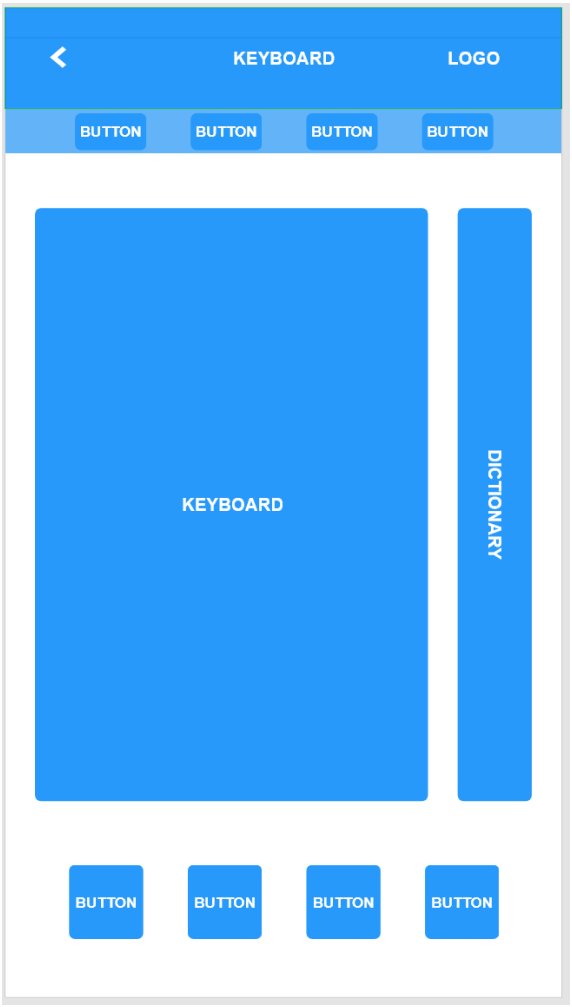


Figure 3: Keyboard wireframe

3.1.3 Framework Selection

Researching frameworks is important during the planning phase. Ending up with a programming language not suited for the project can cause a lot of problems and frustration during development. The bachelor team spent a lot of time researching the pros and cons of different frameworks like React Native, Xamarin and Flutter for the mobile application. To find the best application the group looked at support, features and documentation in each of the frameworks. The group also looked at how easy they were to learn as we had limited experiences with any of them. The group found that Flutter would be the better choice for the application we wanted to make. More reasoning behind this choice can be found in chapter 5.3.1.

Flutter had support for creating desktop applications when the group started the project, but since the framework was only in alpha it was decided to find another framework for the desktop application. The group could not find a lot of frameworks for desktop applications, only some .NET GUI frameworks that we realized only were for the Windows platform. Since we wanted a cross-platform application we had to keep looking until we found Electron. Chapter 5.3.3 explains this choice further.

3.2 Development Methodology

Choosing the best development methodology for a project ensures that it progresses smoothly. Agile methodology is a common approach to software development compared to more traditional models like the *Waterfall Model*. One of the reasons it is used is because of the flexibility it offers. Agile software development is further explained in 3.2.1.

3.2.1 Agile Software Development Methods

Agile methods is an umbrella term for several methodologies. What these methodologies all have in common is that they are iterative processes. Agile methods is a people-focused approach to software development. The approach takes into consideration that the world is rapidly changing and the customers may need changes while in the development cycle.

One of the pros of the agile methods, is the flexibility. When a group is using an agile method, it is easier to adapt to changes. A customer can add, delete or change requirements of each cycle as well as shift the priorities. Another advantage of the agile methods is that it provides better communication. When a developer has a closer relationship to the customer, it is naturally that the communication improves. The projects continual feedback loop after each sprint enables better understanding between developer and customer.

Agile methods also focus on creating less defective products. The end product is usually efficient and robust compared to other methodologies. An efficient and robust product is usually the result of the amount of work being put into the development, implementing, testing, and feedback. Another significant part of the success, is the debugging. Since it is expected to have a working demo of what the developers have implemented in the last iteration, it usually results in more debugging and testing early [60].

One of the cons of agile methods is the lack of documentation. As mentioned before, agile methods have an ever changing scope because of the constant feedback loops. The documentation of a particular module could be rendered obsolete because of a minor or major change. Therefore, documentation is not always emphasized on. In addition, changes will sometimes fail to be added to the final documentation.

Another con with the agile methods is that it often lacks predictability. When the methodology is as flexible as the agile methods, it can be hard to predict how much work a project may take. The customer may demand more and more from the developer each time the project is improved. If the project manager is inexperienced, it may lead to mismanaging the project because he/she fails to rationalize user requirements [60].

3.2.2 Scrum

Scrum in Jira was selected as a developmental methodology for this bachelor project. Scrum is an agile development model commonly used in software projects. Scrum was first introduced on 1995 by Jeff Sutherland and Ken Schwaber. The scrum method is used to develop, deliver, and maintain complex products. The philosophy behind scrum is that knowledge comes from experience, where the developer make decisions on what they know. Scrum is a leading methodology for software developing.

Scrum consists of three basic roles, which together form a Scrum team. Firstly, it is the product owner. The product owner is usually the customer and decides what the product should be. The product owner usually have in depth knowledge of the business area and what the end users need. It is also the product owners that decide what functionality that should be prioritized. The project tasks are usually worked out in a product backlog. The backlog is a list that deals with prioritized and estimated project tasks. The second role in a Scrum team is the developers. The developers job is to build the desired product for product owner. The team usually consist of six to nine persons. The developing team is flexible and self-organizing. The third role is the Scrum master. The responsibility of the scrum master is to establish Scrum as defined in the Scrum Guide. The Scrum master help everyone understand the Scrum theory and is accountable for the effectiveness of the Scrum Team .

The project itself is carried out in an incremental manner. The project undertakes the most important and basic functionality and then builds on more complex functionality from there on. By doing it this way, the developers are able to maximize the value of the project. The team also receives continuous feedback from the customer on what works and what needs to be changed. This is done in sprints [43].

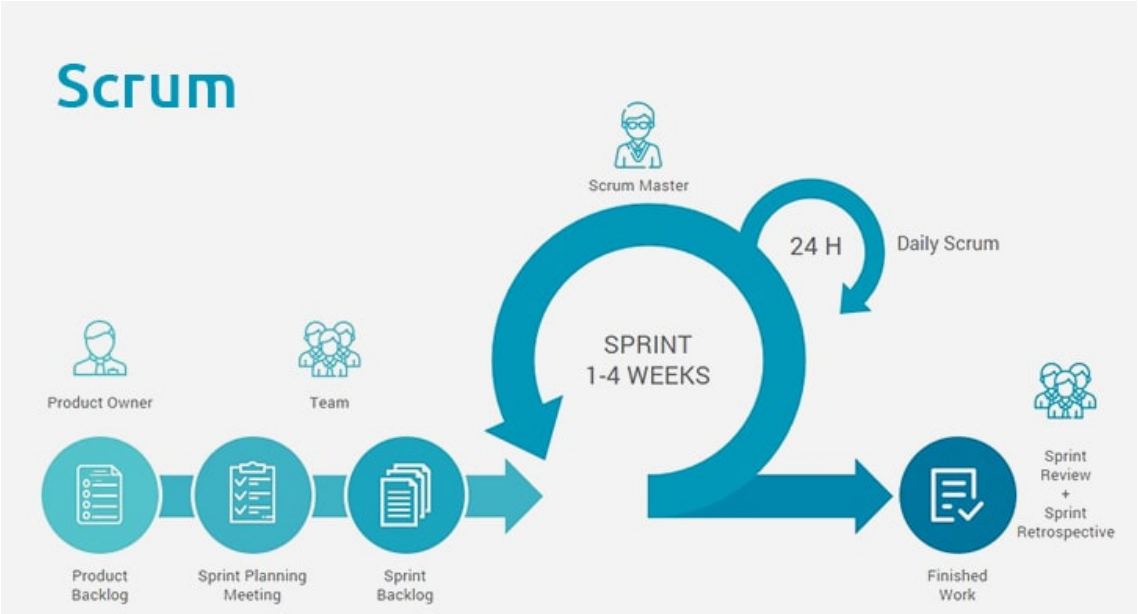


Figure 4: Figure showing the Scrum project cycle. Source: [50]

3.2.3 Sprint

Agile method divides the development process into smaller iterative periods called *sprints*. A sprint is usually between one to four weeks. The goal of a sprint is to develop a demo of the product and the new functionality, and show it to the customer. The customer is able to see the progress of each sprint as well as give feedback on the product. Then, based on the feedback given by the customer, a new sprint can be planned and started. By having short work-iterations the risk of spending a long time on a process that fails because of a mistake early on is reduced. It also improves the chances of the customer getting the product they wanted [4].

A Scrum sprint starts with the planning meeting. The Scrum team determines what is to be delivered when the sprint is done. For a good result it is recommended that each aspect of the functionality is thoroughly reviewed with the product owner and the developers. This is to ensure that the entire development team has a good understanding of both needs

and objectives. When functionality is figured out, the functionality is divided into smaller more detailed work tasks in the backlog. When the sprint is done it is time for the sprint retrospective meeting. Here the team looks back at the sprint and tries to identify what went well and what could have gone better. The team then arrive at a plan to improve the way they work [4].

3.2.4 Weekly Standup

The weekly meeting deals with a quick recap of all the plans, further work and problems that have arisen in the last week. The advantage of weekly meetings is that group members feel accountable for the work done, and that the group is able to reflect on the work. The con of weekly meeting is that is takes up time which could be used on developing the product.[1]

3.2.5 Extreme Programming

Extreme programming is a an agile development method commonly used in software development. This method was introduced by Ken Beck in the 1990s as an agile method that could adapt to changes from the customer. Extreme programming is split into five phases; planning, designing, coding, testing and listening. The planning phase is when clients introduce the requirements to the developers, and the development team creates a project plan. The next phase is designing the software, which minimizes time wasted from redundancies and late changes while developing. In the coding phase the teams develop software by implementing practices specific to extreme programming, like pair programming and collective code ownership. Testing includes both automated testing to check features and customer acceptance tests where the customer confirms that the initial requirements are upheld. Lastly in the listening phase customers and developers discuss what is expected from the project [3].

When using extreme programming it is important to note the set of specific practices that separate it from other development methods. Some of these practices are pair programming, on-site customer, continuous integration, and collective code ownership. Pair programming is the practice of having two developers work on the same code, where one

develops code and the other reviews it and gives feedback, to increase the code quality. The on-site customer practice is when the customer participates a lot during the development to answer questions and decide what should be prioritised. Continuous integration is the practice of continuously committing code throughout the day to keep the team up to date. When the entire team has access to, and can review and update the code in the full system it is called collective code ownership, because the responsibility is on the whole team [3].

3.3 EEG Headset

An EEG headset is a tool used to record the electrical activity of the brain. By placing electrodes along different parts of the head the sensors in the EEG headset are able to measure the potential difference between these electrodes. While the main use of EEG has been in medicine, the introduction of portable EEG headsets have made research using EEG as a computer-brain interface more available. We were provided with Emotivs headsets Emotiv Epoc+ and Emotiv Epoc X, which both use 14 electrodes to measure brain activity. These are headsets intended for research and personal use, and comes with software and an API to make development for the platform easier [17].

Figure 5: An illustration of the EEG headset in use.

3.4 Emotiv

Emotiv is a privately owned bioinformatics company that focuses on EEG and using EEG with BCI. Their main focus is to use EEG to better understand the brain and how it works. To do this they have created EEG-headsets and software [15].

3.4.1 Emotiv App

Emotiv App is used to connect desktop applications, like EmotivBCI or a third-party app like our desktop application, to the EEG headset. Emotiv App also keeps track of updates

on the headsets and on other Emotiv applications like EmotivBCI and Emotiv Pro.

3.4.2 EmotivBCI

EmotivBCI is an application developed by Emotiv that is used to interpret the output from their EEG headsets. EmotivBCI enables the user to train mental commands to control machines with their mind. It also allows the user to view their real-time Performance Metrics, Facial Expression and Motion Sensor data stream from their headsets. The application allows users to create training profiles. By connecting to the Emotiv Cortex API, training profiles that are saved in the cloud can be retrieved by other applications.

EmotivBCI offers guidance and feedback for training Mental Commands. The user is able to create different commands on their training profile. Some examples of different Mental Commands are Push, Pull, Left, and Right. The user is able to connect a different thought to each of this commands. Emotiv recommends thought that are easy to replicate, such as the sour taste of a lemon. When the user has connected a thought to a command they are able to train that command. By training the command, the system is able to notice slight differences in the though and learns to trigger it anyway. After each training, the user receives a score of how good their training was. This can guide the user to better trigger the Mental Command.

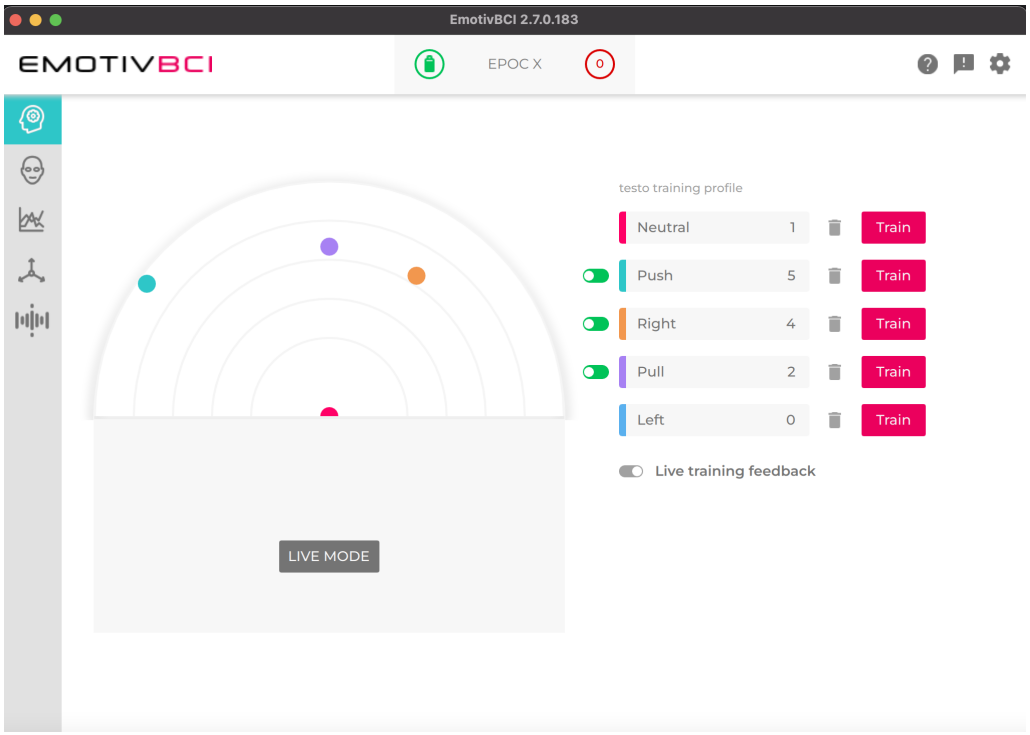


Figure 6: A picture of the mental commands training page.

EmotivBCI also offers feedback when training the facial expressions. Some of the facial expressions that are included are Blink, Left and Right Wink, Surprise, and Smile. The expressions, Surprise, Frown, and Smile can be trained using a machine learning model provided by Emotiv. Blink, Wink-left,Wink-right are already trained by Emotiv. Using these trainable expressions will reduce the chance of the system misinterpreting what expression the user is trying to perform. The application offers a live avatar demo that lets the user see if the training is successful [16].

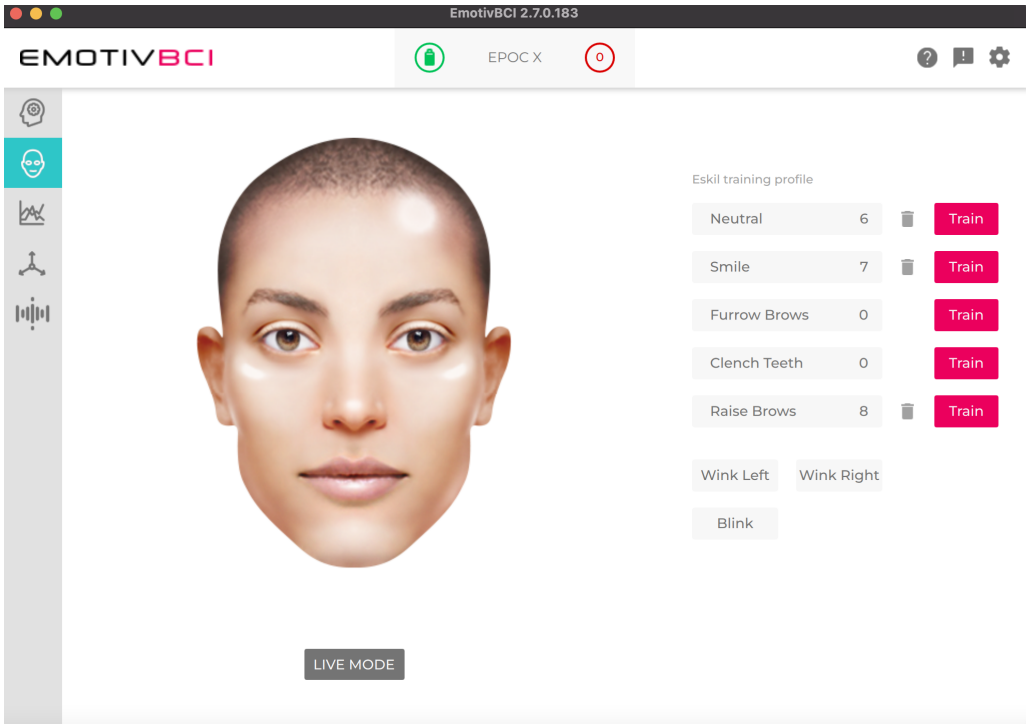


Figure 7: A picture of the facial expression training page.

3.4.3 Emotiv Cortex

Emotiv uses an API based on JSON and WebSockets called the *Cortext API*. Using the Cortex API, developers are able to create third party applications to communicate with the headsets. The API is fully supported on Windows and macOS, as well as being in beta for Ubuntu, iOS and Android. It also supports all of the headsets Emotiv currently has released [10].

3.5 Software Architecture

Software architecture describes the foundational structure of a system. There are several ways to visualize the software architecture of a system. Some examples are use-case diagram, deployment diagrams, and sequence diagrams.

3.5.1 Use-Case

The use-case diagram gives a high level overview of the combined system the group created for the project. The diagram shows how different users with different roles may interact with our system. Due to the high level design and low technicality, a use-case diagram is something you would show to the client to visualise the intended use of the system. For our project we have two users with different roles. The auxiliary user interacts with our desktop app to connect with the Emotiv software and fix the user setup. This allows the user to interact and operate our mobile app using the Emotiv headset.

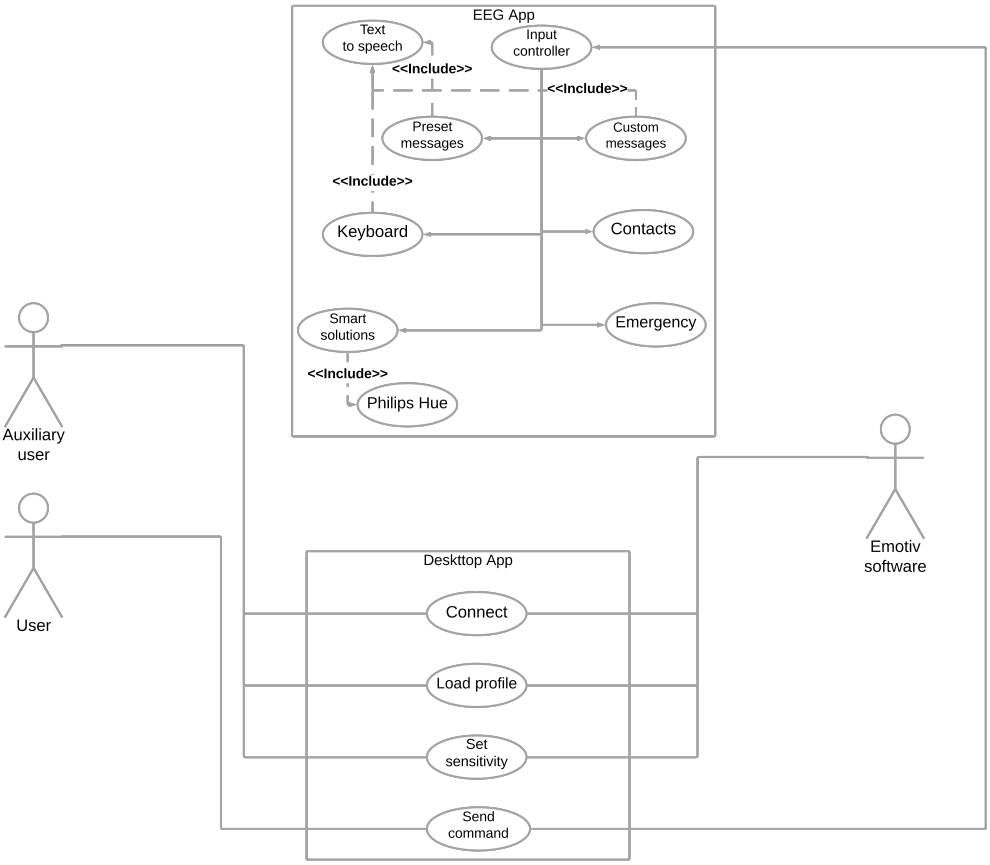


Figure 8: Use-Case Diagram for the full system.

3.5.2 Deployment

Our deployment of the system requires several pieces of hardware and software to work together. First we have our computer, which is the main component that sends and receives information from the other parts of the system. The desktop application we created receives information from the Emotiv headset and sends these commands to the mobile

application. The desktop application sends all requests to the Emotiv App that is installed on the same device. This app listens to requests on port 6868.

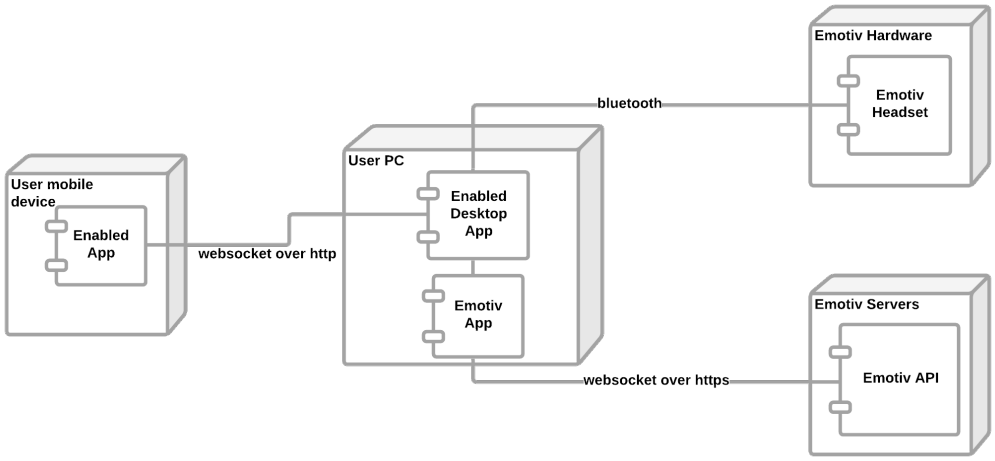


Figure 9: Deployment Diagram for the full system.

3.5.3 Sequence

The sequence of events happening when the user press the connect button after opening the desktop application. When the button is pressed the system will try to connect to the headset. If the Emotiv app does not return any errors, the desktop app will allow you to go to the next page. If there are any errors the desktop app will render errors and the most common way to fix them.

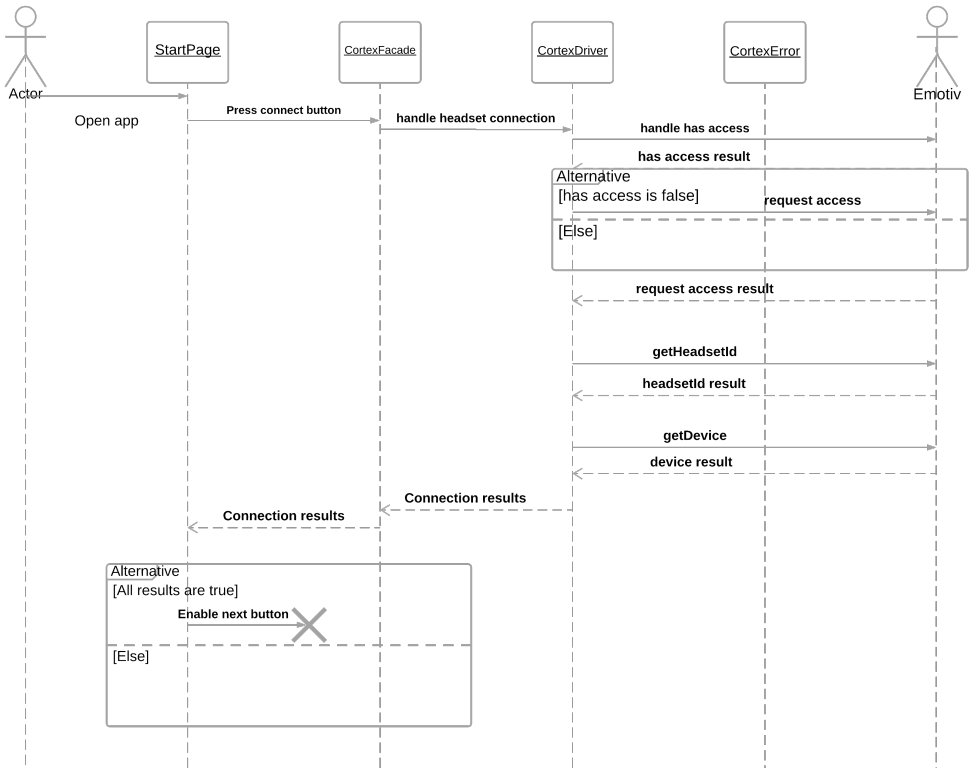


Figure 10: Deployment Diagram for the full system.

3.6 Programming Languages

There are many different programming languages out there for different purposes. Dart is a popular language for developing mobile applications, while HTML, CSS, JavaScript, and Typescript are all popular languages used to create desktop and web applications.

3.6.1 Dart

Dart is an object-oriented programming language. It is developed by Google, and is usually used to create frontend applications, such as web and mobile applications. However, Dart can also be used as a backend language. Similarly to Java, Dart is type safe and also has null-safety. This protects the developer from NullPointerException at runtime. The syntax also resembles Java and C style syntax. Thus, developers moving from Java and C languages will have an easier time learning Dart. Dart also offers a large set of libraries which contains most of what developers need. This includes a HTML library for web applications, a math library, and a conversion library that encodes and decodes JSON

[12].

3.6.2 JavaScript

JavaScript is a popular scripting language. It is often used in combination with HTML and CSS to create interactive web pages and update them dynamically. It can be used to display timers, change color of elements, or show and hide information when interacted with. There are a lot of popular frameworks for JavaScript to make it easier to create advanced web sites. A few examples are React, Angular and Vue. Even though it is mostly used frontend, JavaScript can be used to create web servers using Node.js [33][51].

3.6.3 Typescript

TypeScript is a programming language similar to JavaScript, except it adds static type definitions. All JavaScript code is valid TypeScript code, meaning that TypeScript is JavaScript with some extra features. There are several advantages of using TypeScript. First of all, it adds a type system to JavaScript. It also adds an error-checking feature that generate compilation errors, which JavaScript does not have. TypeScript includes concepts like classes and interfaces, meaning it has support for object oriented programming [65][41].

3.6.4 HTML

HTML is an acronym for *Hyper Text Markup Language*, which is the most common markup language used when creating web pages and applications. It was designed in the 1980s by Sir Tim Berners-Lee, who is often called the inventor of the World Wide Web [64].HTML is used to structure different elements, like titles, paragraphs or links on a website. It is usually combined with CSS and JavaScript in web pages to add functionality and formatting, but can also be used alone to create simple pages [24].

3.6.5 CSS

Cascading Style Sheets, usually called CSS, is used to style the structure created with HTML. CSS tells the browser how to display the elements in the HTML document. It is used to change fonts, colors, sizes and much more. Web pages can be styled directly in the HTML document as well. However, there are a lot of advantages using CSS. CSS has a lot more attributes to specify than HTML, it is faster using one CSS file where equal elements share the same style, and it is easier to change themes on the entire web page [40].

3.7 Frameworks, Software Development Kits and External Libraries

One of the requirements for our project was that the mobile application should be able to run on both Android and iOS devices to make it available for a larger audience. This made the group look at possibilities to run the desktop application on multiple operating systems as well. The group ended up using the Flutter SDK for our mobile application and the Electron framework for the desktop application.

3.7.1 Flutter

Flutter is an open source software development kit made by Google. It is used to create Android and iOS applications simultaneously. Flutter was originally introduced in 2015, but it was not until late 2018 it got out of the alpha stage and was officially launched. Flutter uses Dart as a programming language. Dart compiles the code directly into native code ahead of time. This improves the communication between Flutter and the platform as well as lowering the startup time of the application. After its release, Flutter received a lot of attention from mobile developers, much thanks to the hot reload functionality. Changes in the code are instantly applied in real time, as long as the application is running. This allows the developer to review changes without having to restart the app [56]. A lot of applications are created using Flutter including apps like eBay Motors [58], Google Stadia [9], and My BMW [25].

3.7.2 React

React is a JavaScript library created by Facebook. It is used in frontend development to build interactive user interfaces. Similarly to the widgets in Flutter, React uses components to build the user interface [30]. Several large and well-known applications are created using ReactJS, such as Facebook and Netflix [37].

3.7.3 Electron

Electron is a framework used to create desktop applications using web technologies like HTML, CSS and JavaScript. The framework was originally created by Cheng Zhao in 2013 to make cross-platform development easier using the text-editor Atom, and is now maintained by GitHub. Electron combines Node.js and Chromium to create applications that run on Linux, Mac OS and Windows [14]. Several well-known desktop applications are created using Electron, among these are apps like Visual Studio Code[59], Discord[38], and Slack [8].

3.7.4 Material UI

Material UI is an open source library released in 2014 consisting of React components. Material UI implements Material Design by Google, which is used in software like flutter. It is one of the most used React libraries and is used by several large companies like Amazon, Netflix and NASA. Material UI gives the developer the power to quickly add and customize complex components. The components of Material UI encompasses a lot of tools a designer want when creating a graphical user interface, with components in categories like layout, inputs, navigation, feedback, data display and more [39].

3.7.5 Node.js

Node.js is a JavaScript runtime environment. It is used to execute JavaScript code in a server-side application. Ryan Dahl created Node.js in 2009 and built it on the V8 JavaScript engine for Chrome. After the release of Node.js, developers can create both frontend browser applications and client-side server applications without having to learn a new lan-

guage [47]. A couple examples of business that uses Node.js are Netflix[63], eBay[49], and PayPal[18].

3.7.6 Chromium

Chromium is an open source browser project created by Google in 2008. Multiple popular browsers are based on Chromium, including Google Chrome and Microsoft Edge[34]. It is also used in Electron alongside Node.js to create desktop applications using web technologies.

3.8 Project Management

Project Management is the process of planning, executing and controlling a project to achieve the project goals given. The group utilized different project management products throughout the project. The products used are Atlassian Jira, Atlassian Confluence, Git, GitHub, and Lucidchart [31].

3.8.1 Atlassian Jira Software

Jira Software is a web application created by Atlassian to help teams keep track and manage a project. The software is usually used to create Scrum or Kanban boards to plan and track the development process. It is also possible to make a roadmap and retroactively look at and analyze sprints [7].

3.8.2 Atlassian Confluence

Confluence is also a web application created by Atlassian and can be used to manage a project. However, Confluence is more used as a wiki where team members can share information. Some examples are product requirements, sprint reports or even documents of the project that they work on in Jira. Confluence is often used in combination with Jira on projects as both can be integrated with each other [6].

3.8.3 Git

Git is a free and open source distributed version control system. Git was created by Linus Torvalds in 2005. Git is usually used to coordinate work among programmers during a software project. Speed and data integrity are some of the goals of Git, as well as supporting distributed workflows. [13].

3.8.4 GitHub

GitHub is provider of version control, using Git, for software development. GitHub offers external git repositories for the users. GitHub provides access control, bug tracking, feature request, task management and wikis for every project. These features are included with Github for free. Because of this, Github is often used to host open-source projects.[22]

3.8.5 Lucidchart

Lucidchart was created in 2008 by Lucid Software Inc. It is a web application for creating charts and diagrams. It has both a free tier and a paid subscription [35]. This is a software that we have used in earlier courses during our studies. It is a powerful tool with several pre-made shapes for all of the diagrams and charts we needed for our project. It allows for fast editing of size and colour across single objects and grouped objects.

4 Results

This coming section will include results from the project and comment on how this solution follows the concepts and principles that are mentioned earlier in the report.

4.1 Full System

Our system uses the Emotiv Epoc headsets to get signals from the user. These signals then gets processed in the desktop application. After being processed in the desktop application, it is sent to the mobile application. This enables the user to control the mobile application by using brain activity, which was one of the objectives of this thesis. The commands have been trained in advance using the EmotivBCI application. In the mobile application we have four controls that can be used - *Push, Pull, Right, and Left*. These are used to navigate and select the different options possible in the application. Figure 9 shows the structure of the full system and how the different parts communicate.

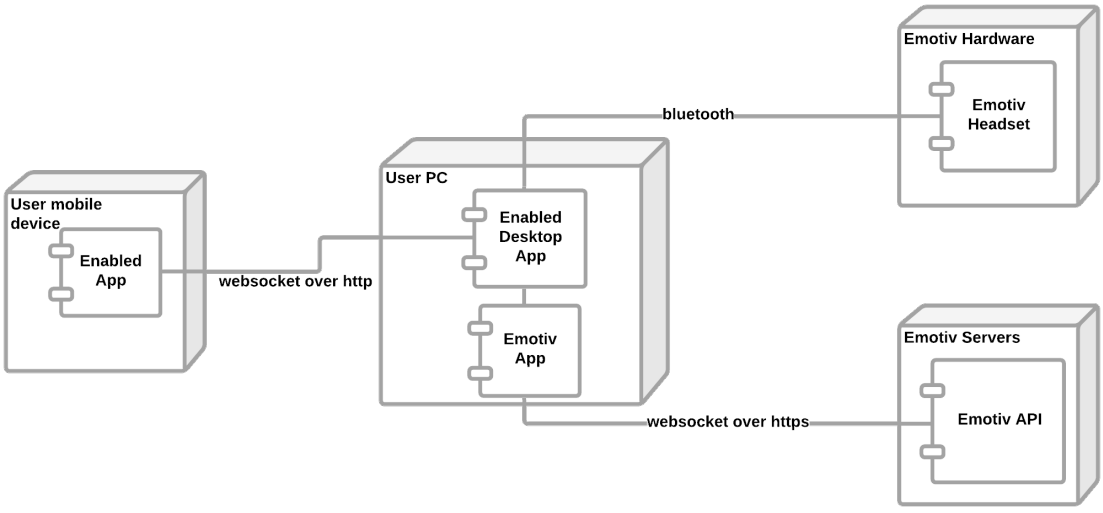


Figure 11: Structure of the entire system, including phone, desktop computer, Emotiv servers and EEG headsets

4.2 System Architecture

Our system uses software and hardware provided by Emotiv. This restricts some of the architectural design choices that we could make during our thesis. This is because

EmotivBCI handles profile training, headset-data parsing etc. EmotivBCI communicates with the desktop application through a local port, and the headset communicates over Bluetooth. The computer is the main part of the system. It handles all the data from the Emotiv software and hardware, and sends this data to the mobile application in form of commands. Enabled EEG and desktop application communicates these commands over http. The Enabled app then interprets these commands and converts them into interactions.

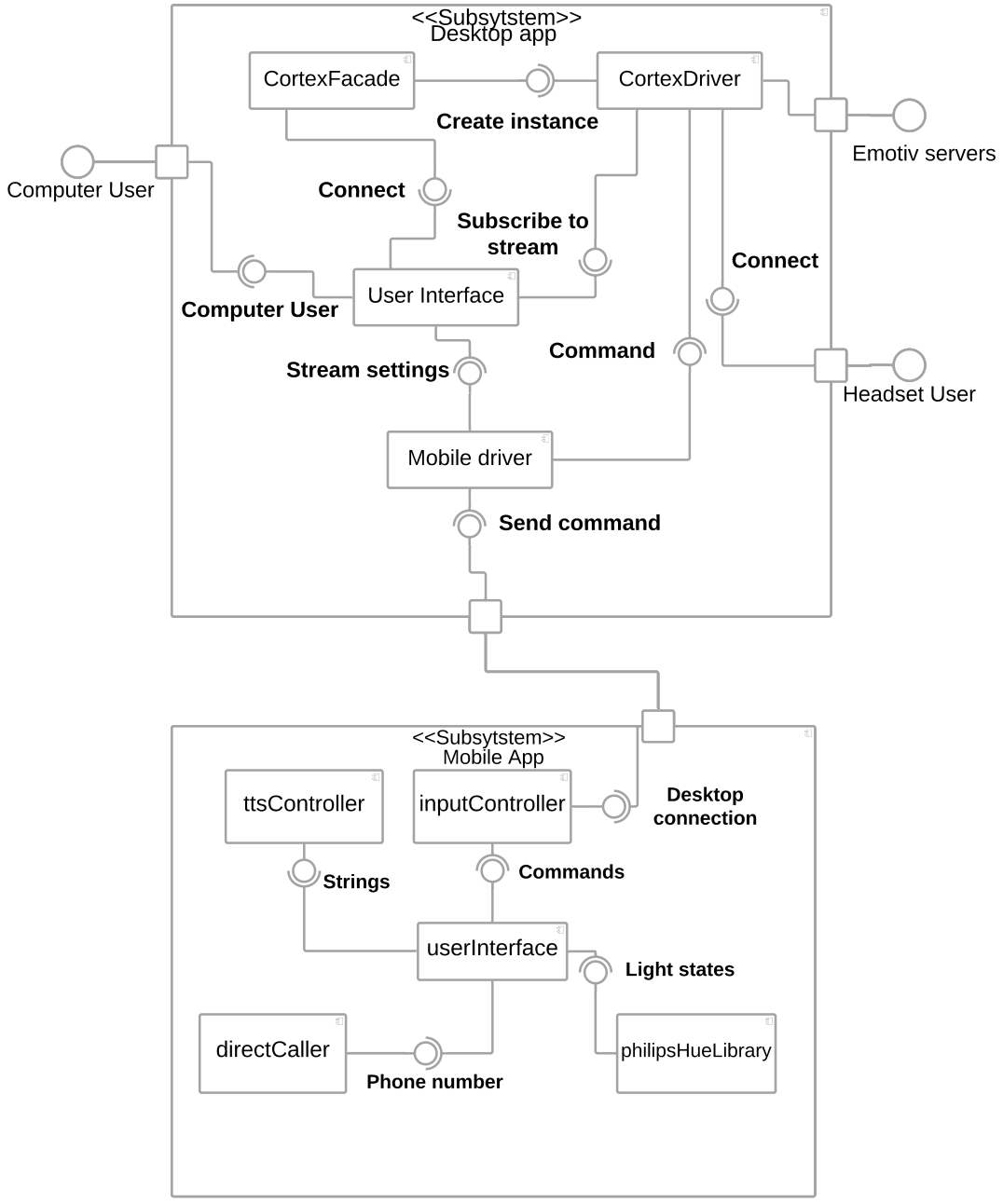


Figure 12: Component Diagram for the full system.

4.3 Desktop Application

A goal for this bachelor was to develop a desktop application that runs on both Windows and macOS. The group is using hardware from Emotiv and needed a way to send the data from the headset to the mobile app. Emotiv offers a solution to this through their EmotivBCI application. This application has a socket server listening that enables the group to send request and retrieve information about the headset in real-time. The Enabled desktop application is developed for both Windows and macOS, which was on of the goals for this project.

4.3.1 User Interface

For the desktop application the group chose a simple user interface. The responsibility of the application was split into four pages. These pages were the setup, profile selection, add IP and headset stream page.

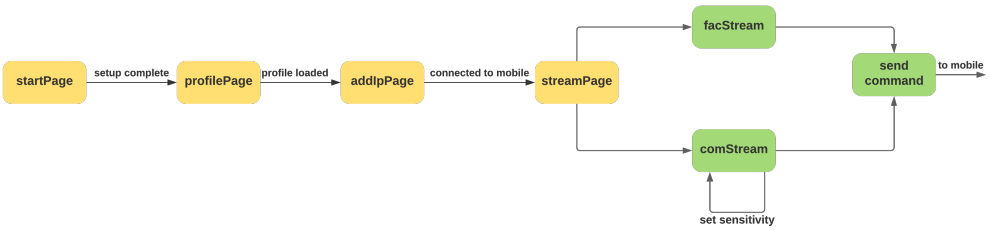


Figure 13: A state diagram of the desktop application.

Start Page

The first page is the setup page. The Emotiv hardware and software require setup. Some examples are checking if the application has access to the EmotivBCI app, if the headset is correctly connected and if the client-secret and username is correct. The responsibility of the start-page is to check these errors. If everything is correctly configured, the page displays a success-message and a green check mark. If something went wrong it displays an error message. The error message is meant to help the user find where their setup went wrong and how to fix it.

Figure 14: The start page.

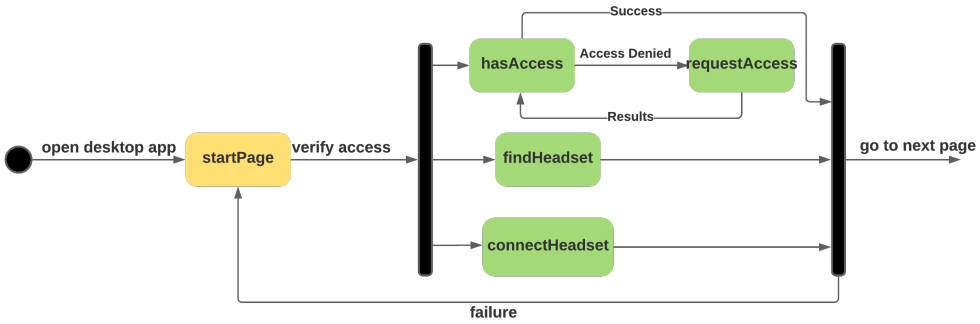


Figure 15: The state diagram of the setup page.

Select Profile Page

The second page is where the user selects their Emotiv training profile.

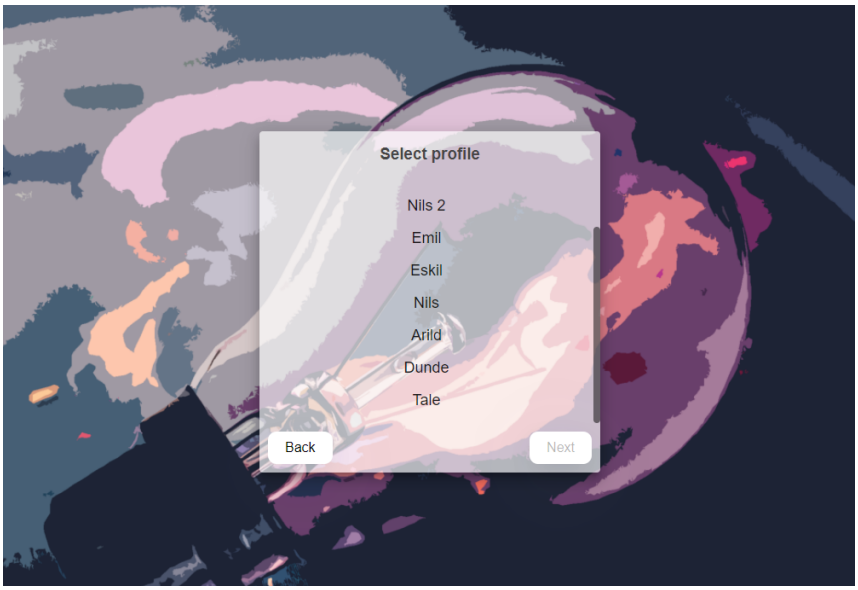


Figure 16: A page to select your training profile.

Add Phone IP Page

The third page is the page where the user adds their phones IP address. On this page the user can find a link labeled “find IP address” and an input field. The link opens a modal dialog. The dialog shows the user how to find the IP address of their phone, and how to utilize the Enabled mobile app to find the IP address.

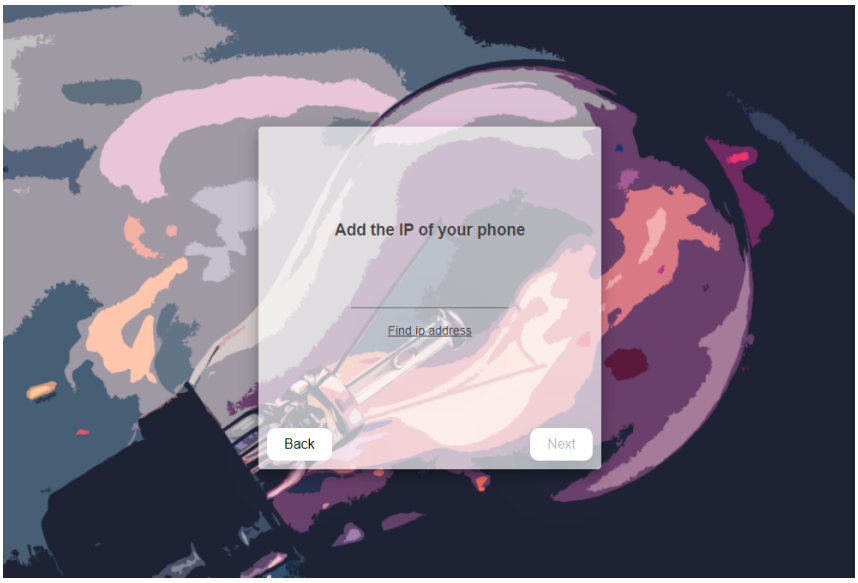


Figure 17: A page to add the IP address of the phone.

Stream Page

The final page is the stream page. This page has two tabs the user can toggle between. The first tab is the *Mental Command* tab. When this tab is selected, the user is able

to use the mental commands from their training profile. Another feature on this tab is that the user can set the sensitivity of their headset. The tab also tells the user which Mental Commands that are supported in the mobile app. The other tab is the *Facial Expression* tab. This tab starts the Facial Expression stream and shows the user which Facial Expression commands that are supported in the mobile app.

Figure 18: The stream page.

4.3.2 Applying Don Norman's Design Principles to the Desktop Application

When the group began to design the desktop application it was discussed how to apply Don Norman's design principles to the application. The group wanted the desktop application to be simple and intuitive for the user. With this in mind, the group began to apply Don Norman's first principle, *visibility*. The group wanted the applications functionality to be clearly visible and decided that all functionality should be inside a main display.

Secondly, the group wanted to focus on the applications feedback. This was to enhance user-experience. The group began to sort out the different errors that could arise from the EmotivBCI application. When an error occurred, the application would alert the user of the reason for the error and how to deal with it. The group tested the system thoroughly and found the most common errors that occurred in the system. Even though it was well tested, the group still experienced some errors, which were not documented in the Emotiv API. The system would prompt a default error message if this were to occur.

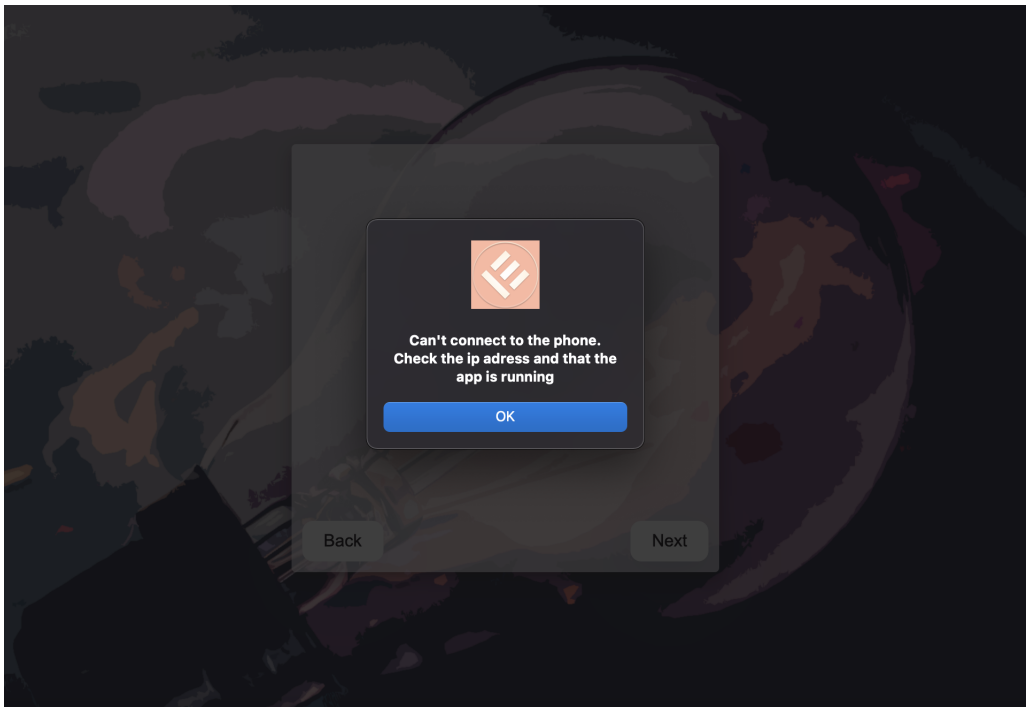


Figure 19: The alert when the system is not able to find a mobile socket server.

As the group progressed, it was noticed several delays when the application fetched data or tried to connect to a socket. The solution to this problem was to give the user some form of feedback, to show that something was happening. It was decided to implement a *circular progress bar* for the select profile page, the add IP address page and stream page. This would show the user that something was loading.

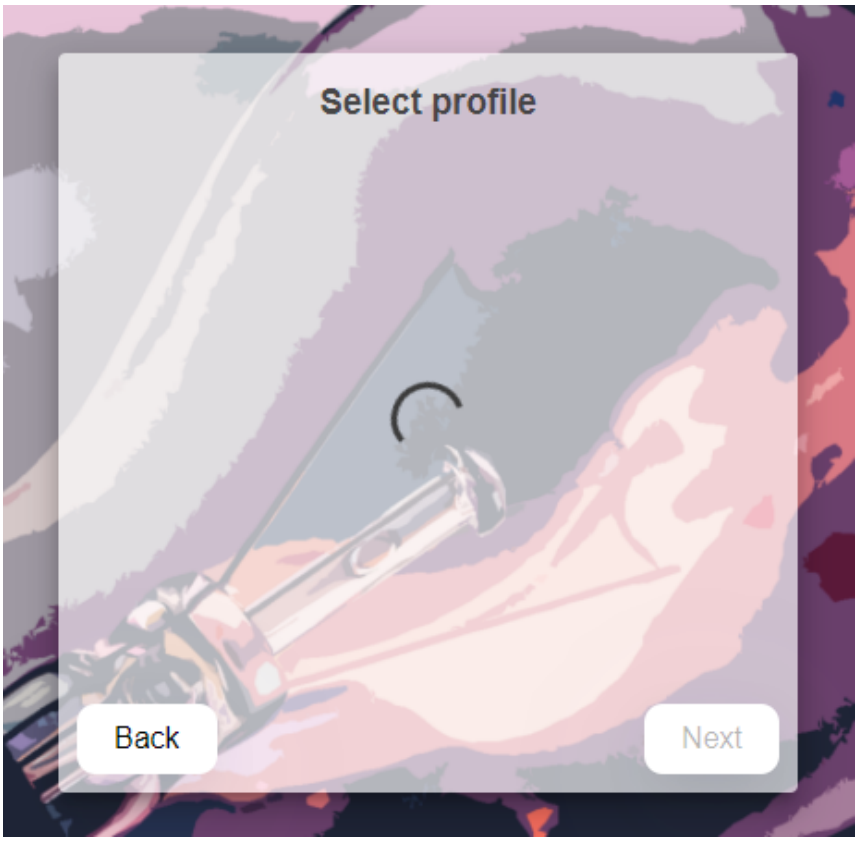


Figure 20: The loading circle on select profile.

Another great example of the feedback and constrain principle implemented in the desktop application is the next button. The button is shown in 4.3.1. The button is grayed-out when you first navigate to the page. After the required action has been accomplished, the buttons changes to the regular color. For the start page it is when you have connected properly. For the select profile page it is when you have selected a profile and for the add IP address it is when you have entered a valid IP into the input field. The group also applied the consistency principle by keeping the navigation buttons consistent across all the pages in the application.

Finally, the group applied the mapping principle to the application by implementing scroll-bars where the content had an overflow. This discloses where the user currently is in the list, and that there are more available options or information that can be reached by scrolling.

Figure 21: The scroll bar on select profile.

4.3.3 Applying Design Pattern to the Desktop Application

The group were to develop a driver class to communicate with the Emotiv Application. The group had limited knowledge about socket programming and began to test the example provided by Emotiv [28]. As we ran the example code, the group noticed that their socket server only allowed for one connection at a time. To avoid any cleanup errors, it was decided that the driver should implement the singleton design pattern. As a result, it was only possible to create one instance of the class and only one socket connection at a time. This solved the problem with only having one connection at a time.

Listing 1: How the singleton pattern is implemented in the CortexDriver

```
/**
  This class works as a connection between an app and the Emotiv API.
  This class uses async/await and Promise for request and needs to be run
      on sync.
 */
```

```
class CortexDriver {
  private static instance: CortexDriver;
  ...

  private constructor() {
    ...
  }
  static getInstance(): CortexDriver {
    if (CortexDriver.instance) {
      return CortexDriver.instance;
    }
    CortexDriver.instance = new CortexDriver();
    return CortexDriver.instance;
  }
```

The group also noticed that the Emotiv example code had high coupling and low cohesion. Nearly every method called on one or more method, which again called on several methods. It was decided to refactor the code and let the driver class focus on the requests to the Emotive BCI app, and not how to process the response. In this way, each method only had one responsibility, as well as moving the responsibility from the driver class to the class that uses it. Almost every request requires another request to be executed after. To deal with this we decided to create a facade class that could handle the different subsequent requests and process the data received from the requests. This made it easier to use the functions from the Emotiv Cortex API.

Listing 2: An example of how the facade takes a complex task and offers an easy way to use the functionality.

```
class CortexFacade {
  private static instance: CortexFacade;
  private driver = CortexDriver.getInstance();
  ...

  private constructor() {}
```

```
static getInstance(): CortexFacade {
  if (CortexFacade.instance) {
    return CortexFacade.instance;
  }
  CortexFacade.instance = new CortexFacade();
  return CortexFacade.instance;
}
...

  /**
 * Unloads the old profile and loads the new profile.
 * @param selectedProfile The selected profile you want to load.
 * @returns Undefined or a CortexError if an error occurrs.
 */
SetProfile = async (selectedProfile: string) => {
  try {
    if (!this.driver.isConnected()) {
      let connected = await this.driver.awaitSocketOpening();
      if (!connected) {
        return new CortexError(6, '');
      }
    }
    this.authToken = await this.driver.authorize();
    this.headsetId = await this.driver.queryHeadsetId();

    let hasLoadedProfile = await this.driver.hasCurrentProfile(
      this.authToken,
      this.headsetId
    );

    if (hasLoadedProfile) {
      await this.driver.setupProfile(
        this.authToken,
        this.headsetId,
        '',
```

```
          'unload'
        );
      }
      await this.driver.setupProfile(
        this.authToken,
        this.headsetId,
        selectedProfile,
        'load'
      );
      return;
    } catch (error) {
      return this.errorHandling(error);
    }
  };
```

A third problem occurred when the group began implementing the headset stream. The stream reads brain activity from the user and sends it to a socket through an *Message Event*. If another class or instance would like to see the stream information they would have to access the socket and set their own event. This limits the implementation in two ways. First, it will only be available for one instance at a time, because each instance have to set and handle their event it will override the last event configurations. Secondly, it would increase the coupling between the two components. To resolve this problem we decided to implements the observer pattern. The class *CortexDriver* has the role as a subject. We created an interface named *streamObserver*.

Listing 3: The stream observer interface.

```
interface StreamObserver {
  sendCommand(command: FacDataSample | ComDataSample): void;
}
```

The observer class needs to implement the interface and subscribe to the CortexDriver to listen to the stream.

Listing 4: The stream observer interface implemented in the mobile driver class.

```
/**
 * The class connects to a Websockets server and sends the commands it
    recieves to it.
 */
class MobileDriver implements StreamObserver {
 ...
  /**
   * Send a command to the mobile socket server.
   * @param command sends a command to the websocket server
   */
  sendCommand(command: FacDataSample | ComDataSample): void {
    ...
  }
```

The CortexDriver notifies every observer when it receives a new update, This way it is made possible to have several components listen to the stream. Another benefit is that it decouples the two classes.

Listing 5: The subscribe, unsubscribe and notification in the CortexDriver class.

```
  ...
  /**
   * Subscribes to the stream.
   * @param observer The observer
   */
  public async subscribe(observer: StreamObserver) {
    this.observers.push(observer);
  }
  /**
   * Unsubscribes to the stream.
   * @param observer the observer to remove from the array.
   */
  public unsubscribe(observer: StreamObserver) {
    let observerToRemove = observer;
    this.observers = this.observers.filter((item) => item !==
```

```
        observerToRemove);
  }
  /**
   * Notifies all the listeners.
   * @param streamCommand The command from the stream.
   */
  private notify(streamCommand: FacDataSample | ComDataSample) {
    this.observers.forEach((observer) =>
        observer.sendCommand(streamCommand));
  }
}
```

The group thought that someone else may work and further develop the application in the future. Subsequently, it was decided that the React components should be easy to understand and reuse. As a result, the *React container pattern* was implemented. This design pattern separates data fetching from the presentational components.

4.3.4 Testing

The group chose to integrate the Material UI library into the desktop application. Material UI recommends to test the application without trying to test too closely to the Material UI components. There are two reasons for this. The first reason is that the developer should expect that the components are already well-tested and works as intended. The second reason is that if the group tests too close to the *React component tree*, the tests are prone to fail if the Material UI developer were to makes internal changes to the components.

Therefore, it was decided to test the components that wraps around the Material UI components and the logic we had implemented for them. A good example of this is the input field we have in the add IP page. Instead of looking for Material UI component we are looking for the input component. The group are testing that the user are able to input information into the input field and that our key-events are working as intended. The group focused on testing the visual aspect of the application. That is, the aspects that the user can see.

Listing 6: Component testing of an input component. Tests the keyEvent and that it is possible for the user to type in the textfield.

```
describe('test the custom input component', () => {
  test('The enter key beeing pressed', () => {
    const handleChange = jest.fn( (value) => {});
    const handleKeyPress = jest.fn();
    render(
      <CustomInput
        handleChange={handleChange}
        handleKeyPress={handleKeyPress}
      />
    );

    //fires the enter key press event.
    fireEvent.keyPress(screen.getByTestId('input'), {
      key: 'Enter',
      code: 'Enter',
      keyCode: '13',
      charCode: '13',
    });
    //checks if the keypress event has been fired one time
    expect(handleKeyPress).toHaveBeenCalledTimes(1);
  });

  test('The users is able to write in the textfield', () => {
    const handleChange = jest.fn( (value) => {});
    const handleKeyPress = jest.fn();
    render(
      <CustomInput
        handleChange={handleChange}
        handleKeyPress={handleKeyPress}
      />
    );
```

```
  const input = screen.getByRole('textbox', {name: ""});
   //Checks if the textfield is loaded in the document
   expect(input).toBeInTheDocument();

   //Changes the value of the textfield
   fireEvent.change(input, { target: { value: '1.0.0.1' } })
   //Checks if the value is changed
   expect(input).toHaveValue('1.0.0.1');

   //Changes the value of the textfield
   fireEvent.change(input, { target: { value: '' } })
   //Checks if the text is removed.
   expect(input).toHaveValue('');
 });
```

4.3.5 Emotiv Driver

The group needed a way to communicate and read information from the Emotiv headsets. This is the task of the Emotiv driver class. The driver communicates with the Emotiv headsets and the Cortex API through the EmotivBCI application. The class sends request and receives responses from the API. The class authorize the user, connects to a headset, fetches training profiles, loads training profiles, starts the headset streams and sets the headset sensitivity.

4.4 Mobile Application

The Enabled EEG mobile application is available on both iOS and Android, completing one of the initial goals of the bachelor project. Since our main goal for this project was to make the application have more functionality, our first task was to decide what functionality could improve the application. The group received feedback from the client regarding the functionality, and one request was that he wanted a way to communicate things easily or have preset messages. In the process of choosing the mobile application, we had to

keep in mind that the application was going to be controlled by four inputs and that it needed an easy-to-use approach.

The group decided to create six pages, that is the *Main, Keyboard, Needs, Custom, Contacts and Smart Page*. The group also added a shortcut to call and contact the nurse or a close contact chosen by the user. Figure 22 shows a general overview of the component architecture in the mobile application. The main page ensures the header and bottom navigation bar stays consistent through the application, while the body changes depending on the current page. Consistency like this is considered good design as explained in 2.6.6.

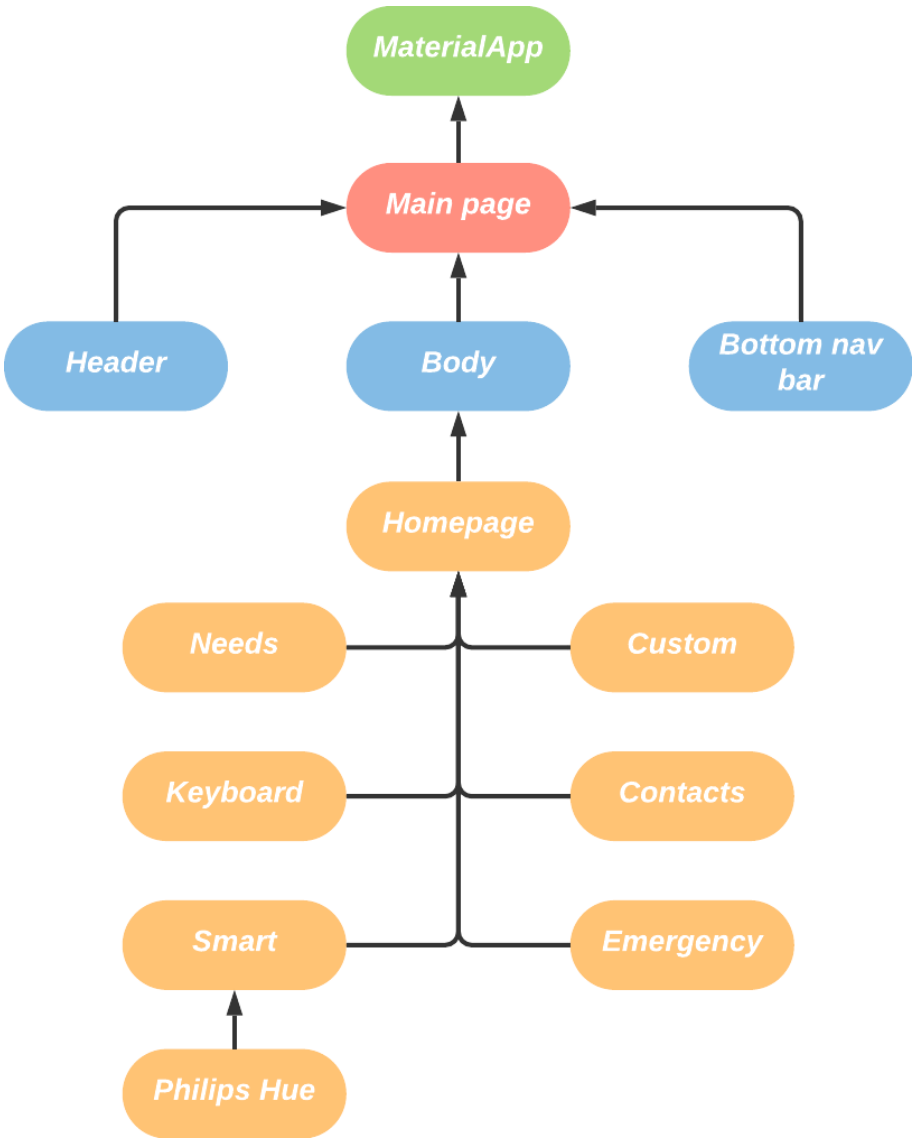
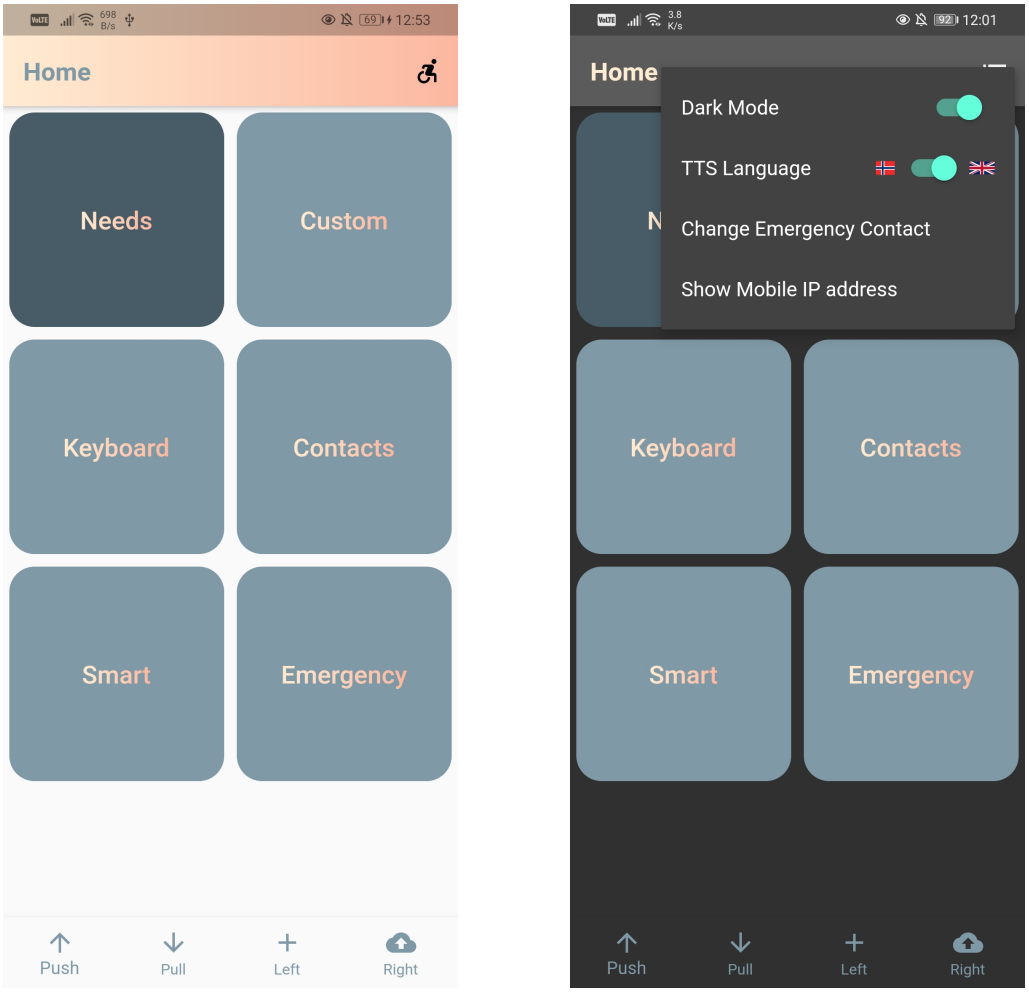


Figure 22: Component architecture.

4.4.1 Layout

Main Page

The first page the user enter is the main page. The user is able to navigate to each of the underlying pages from here and it consists of the six main page buttons that each take the user to their respective pages. The only exception is the emergency button, which directly calls an emergency contact chosen by the user. The main page also contain a drop down menu in the top right corner. In this drop down menu, the user have the option to change the theme of the application, change the emergency contact, switch between Norwegian and English text-to-speech and dictionary, and show the IP address of the phone. This is helpful when setting up the desktop application.



(a) Main page (b) Main page with drop down menu shown

Figure 23: Main page layout

Keyboard

The keyboard layout used in our application is the same as the one from the 2017 bach-

elor (2.3.2). The previous keyboard used two inputs from the headset, while we wanted to utilize four. The group determined that four inputs would make a faster and easier navigation. With four inputs the user are able to navigate both *Up and Down* and *Right and Left* at the same time. This makes it easier to correct mistakes. The keyboard page also has a dictionary, which allows users to quickly select words and form sentences. The dictionary uses a Norwegian and English frequency lists to build a selection of the most likely words, and is dynamically updated as the user types on the keyboard.

The old bachelor keyboard did not have any feedback when a message was sent from the keyboard. We decided to improve upon this, and add text-to-speech when a message was finished.

The Keyboard page fulfills three of the primary objectives for this project, by providing opportunities for the application users to communicate with others, providing text-to-speech functionality, and providing a keyboard designed for typing with brain signals.

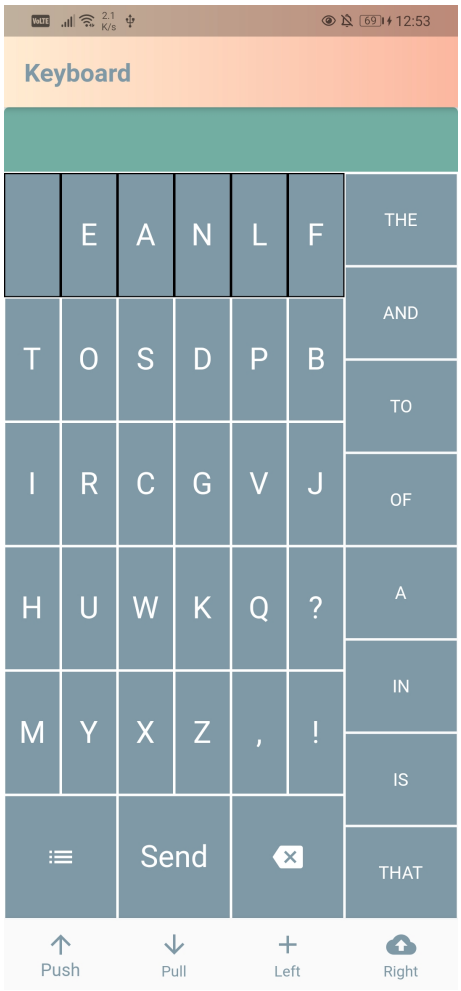


Figure 24: Keyboard page layout.

Needs and Custom

The Needs and the Custom pages have preset buttons or an option to add interactions and necessities. In the custom page, the user has the option to add custom messages, such as phrases or words the user often use. The user is also able to add the phrases and words to different custom categories.

Two main objectives are completed in the Needs and Custom pages as they provide opportunities for the application users to communicate with others, and add text-to-speech functionality.

(a) Needs page (b) Custom page

Figure 25: Custom and needs page payout

Contacts

The contact page enables the user to add new contacts to the list and call them instantly using the application. It is also possible to remove contacts that no longer are in use, but the user have to get some help from a nurse or another person.

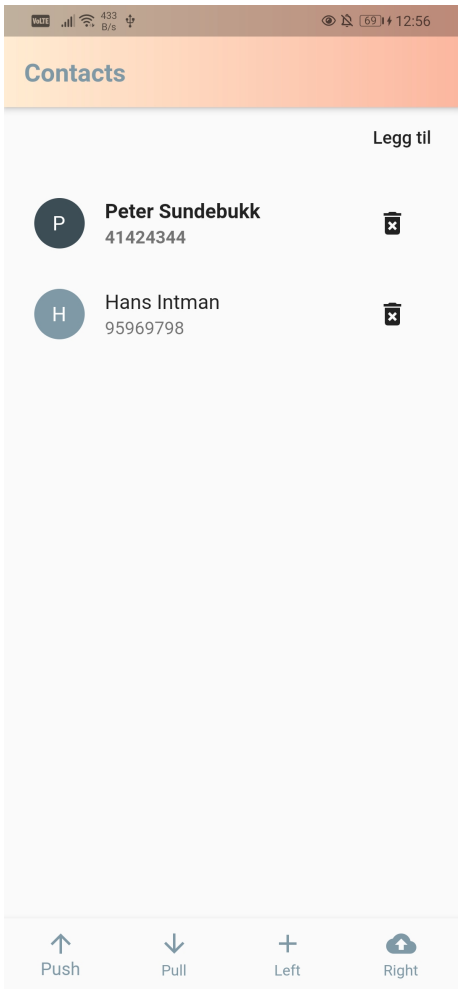


Figure 26: Contacts page.

Philips Hue

The final page is the Smart Page where the group added the Philips Hue page. The user have the option to control the state of their Philips Hue smart lights by turning them on/off, dimming/brightening, and changing the theme of the light states. One of the initial goals was to add some sort of smart solution and integrate it into the mobile application, this is done by integrating features to control Philips Hue smart lights.

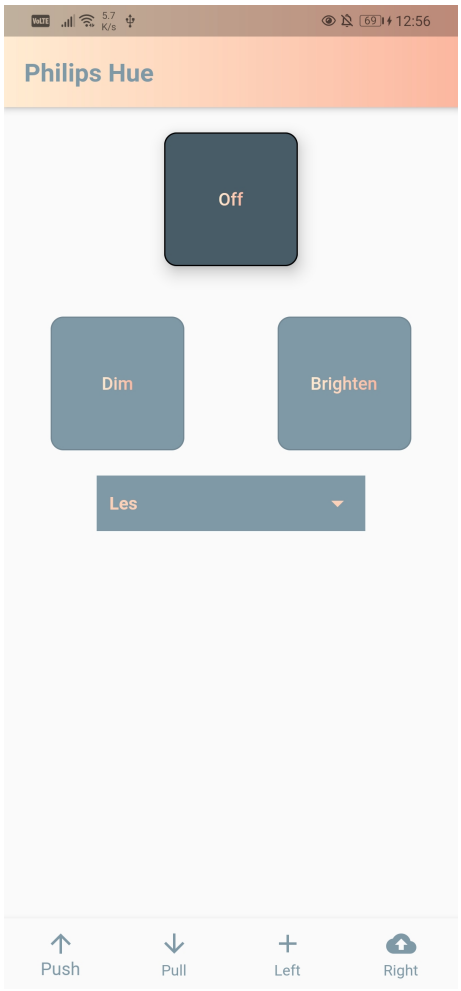


Figure 27: Philips Hue Layout.

Emergency Contact

In addition to the five pages, the group wanted a quick way to contact a nurse or emergency contact. The group decided that this should be a button on the main page, which calls the contact directly when pressed. The emergency contact is added in the drop-down menu or the first time the button is pressed. In addition, if no emergency contact is set and the emergency button is pressed, a popup opens up to add an emergency contact as shown in figure 28.

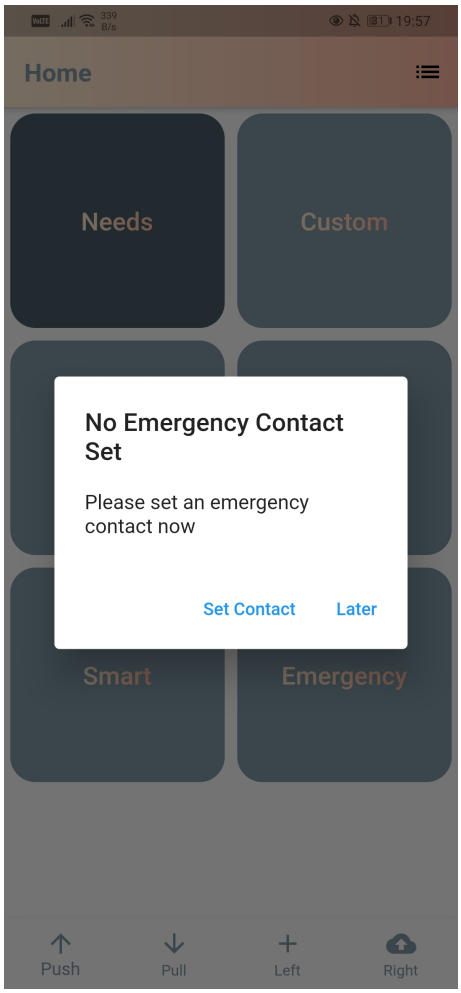


Figure 28: Requests an emergency contact if no contact is set

4.4.2 Applying Don Norman's Design Principles to the Mobile Application

The first principle the group focused on was consistency. The group wanted both the app design and the controls to be consistent and not deviate too far from each other. The first application of this design principle was to implement a design concept. The design concept was created to ensure that all group members had a "guideline" to follow when they created the application pages. This was meant to give the application a consistent, yet professional feel. A second action the group took to apply the consistency principle, was to implement an overall controller to the application. This resulted in all of the controls to act universal to the same kind of inputs, for all the different pages as shown in figures 23a, 24, 25, 26, and 27.

Feedback was also something the group wanted to improve on. Making sure the way we did this was to make the button currently selected a darker color so you could see where

you were and navigate through the app easier. By applying a darker color on the selected button, the user would not question where in the application they currently were and there would be no confusion. This principle was also implemented through the use of text-to-speech in the application. When a button is pressed in the Needs or Custom page the user receives feedback through text-to-speech.

A third principle implemented in the app is Don Norman's constraints principle, described in chapter 2.6.5. This principle is implemented in the Smart page. If the user does not have a Philips Hue bridge on the same network the button leading to the Philips Hue page is grayed out, indicating that the button is disabled, and that it is not possible to move on. An example of this is shown in 29.

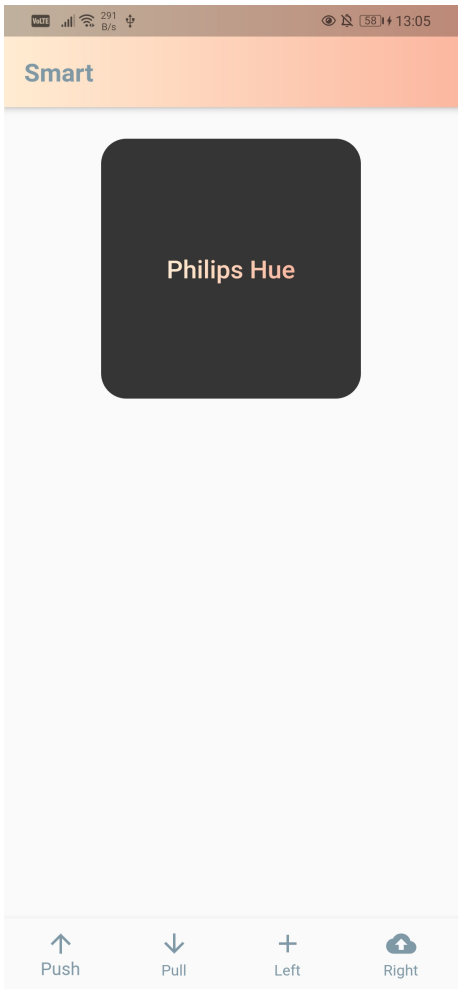


Figure 29: Disabled button on Smart page.

4.4.3 Applying Design Pattern Principles to the Mobile Application

The mobile application is relying on commands from the Emotiv headset to navigate. For this reason it is essential that there is little to no delay from when the command is sent to when something happens on the application. To solve this issue we decided to implement the observer pattern on the stream. We used the *StreamController* class from the flutter SDK. This class enables listeners and updates, which improves performance, and helps to decouple the code.

Another example of applying the design principles to our mobile application is the *HueApi*-class. This class uses the singleton design principle, explained in 2.5.1, because having only one instance of the HueApi class created across the system ensures a consistent connection to the physical bridge. Listing 7 shows how the singleton pattern is implemented.

Listing 7: How the singleton pattern is implemented in HueApi

```
class HueApi {
  static final HueApi _api = HueApi._internal();

  factory HueApi(Client client) {
    _api.client = client;
    return _api;
  }

  HueApi._internal();
  ...
}
```

A third design pattern applied to the mobile application is the facade pattern described in 2.5.3. Facade pattern is applied in the HueApi-class to simplify the API, by removing the need of parameters. Instead of calling on a setBrightness-method with a parameter, the brightnessDown and brightnessUp methods change the brightness by a specific amount, reducing user error. Listing 8 shows how the facade pattern is implemented in the HueApi-class. Another important aspect to note is the underscore in the _setBrightness method. This underscore is what defines a private method in Dart, meaning a method that can not

be referenced from another class.

Listing 8: How the facade pattern is implemented in HueApi

```
class HueApi {
  ...
  void brightnessDown() {
    int brightness = lights.first.state.brightness - 50;
    ...
    _setBrightness(brightness);
  }

  void brightnessUp() {
    int brightness = lights.first.state.brightness + 50;
    ...
    _setBrightness(brightness);
  }

  Future<void> _setBrightness(int brightness) async {
    ...
  }
  ...
}
```

4.4.4 Usability Testing

A usability testing regime was created to test how user friendly and intuitive the mobile application was, and also to see if any unknown bugs were found in the app. The test regime consisted of three simple tests that would test the most frequently used features in the app, and a short interview afterwards to get some feedback from the testers. All tests were measured by how long they took to complete, and if the tester managed to complete it at all. The first test was to navigate to the contacts page and add a new contact to the list. Next the testers had to figure out how to add a new emergency contact, and then call it. In the last test, testers had to navigate to the keyboard page and write a sentence in the text field. These tests were created to test the user friendliness of the application and not

the testers.

4.4.5 Testing

Testing in Flutter is done by implementing the *flutter_test* library. This is part of the test package created by the Dart development team. The conventional way to test widgets, or components as they are usually called in other frameworks, is to tests if the visual aspect of the widget corresponds with the intended design. The components that need new tests are the components that are custom made by the developer. As for components made by other developers or firms, it is assumed that the components already are well-tested and working as intended. Component testing is generally implemented as a tool to ensure each component does what it is supposed to do in the software, as described in 2.7.3.

The tests in the mobile application are therefore only written to test the custom made widgets, such as the home page buttons or the keyboard page. A good example is the test for the keyboard page. The test makes sure that there are no duplicate widgets created. This is to ensure the app runs smoothly and efficiently. Component testing should also test user input. Another good example of this was the test that taps buttons on the keyboard to write a word and then checks the "TextField"-object to make sure the correct word is entered. Listing 9 shows the implementation of tests on the KeyboardPage-widget, and figure 30 shows a successful test result.

Listing 9: Implementation of tests on KeyboardPage

```
//Testing the keyboard page to make sure no duplicate widgets are
//created, and that the keys behave as expected.
testWidgets('Keyboard page testing', (WidgetTester tester) async {
  KeyboardPage page = KeyboardPage();
  await tester.pumpWidget(makeTestableWidget(child: page));

  //Expecting to find only one of each of the widgets TextField,
  //CustomKeyboard and CustomDictionary.
  expect(find.byType(TextField), findsOneWidget);
  expect(find.byType(CustomKeyboard), findsOneWidget);
  expect(find.byType(CustomDictionary), findsOneWidget);
```

```
    ...
    //Testing if TextField gets input when KeyboardKeys are pressed.
    TextField textField = find.byType(TextField).evaluate().first.widget;
    //Expecting empty TextField to begin with.
    expect(textField.controller.text, "");
    //Pressing the keys T-E-S-T.
    await tester.tap(find.byType(KeyboardKey).at(6));
    await tester.tap(find.byType(KeyboardKey).at(1));
    await tester.tap(find.byType(KeyboardKey).at(8));
    await tester.tap(find.byType(KeyboardKey).at(6));
    //Expecting TextField to contain the text "TEST".
    expect(textField.controller.text, "TEST");
  });
```

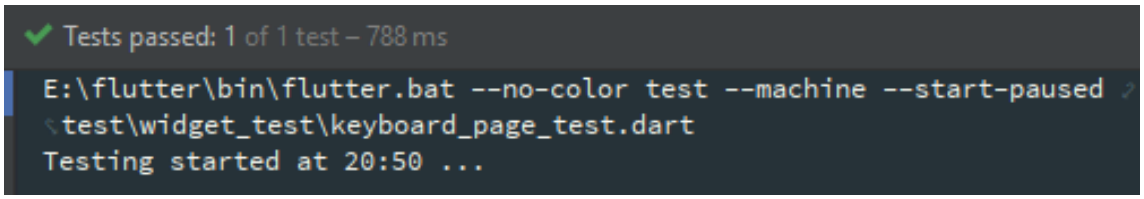


Figure 30: Test of KeyboardPage is successful

4.5 Philips Hue Flutter Library

One of the project objectives were to make the people in need of our application more independent. The group wanted the users to be able to do more, with less help. To achieve that goal, the group realised that implementation of smart equipment, which could be controlled through our application, was the way to go. Accessibility was the main point we had to look at. The group spent some time researching different equipment. It was also required that the group looked at several aspects of the products. The group had to make sure it was possible to implement the product into the mobile application. What could we do with the smart equipment from our app? Was the API open for developers? Did we need a license for it? Those were important questions we needed answered before deciding what to go with. Price was also a point of consideration as we wanted it to be available to most users. After thorough research the group decided on Philips Hue

lighting.

Philips Hue uses an open REST API. The bridge connects to other equipment on the same network and the developer has full access to the API and features of the smart equipment. The Philips Hue gadgets are also relatively cheap and pretty popular. This makes it an affordable and realistic option for most users. After settling for Hue, the group began looking at flutter libraries to control the lights with our app. The group tested and researched different libraries. The conclusion of this process was that there were no libraries with enough documentation or features for us to use. As a result, the group decided to create its own library. There were several benefits by creating a new library. The first benefit was that it ensured the implementation of all the features the application needed. The second benefit was that the group knew how it worked. This made it easier to implement in the application. The end result of the library is a product that automatically connects to a Philips Hue Bridge on the same network. It has methods, shown in the class diagram in figure 31, to get information about lights, scenes and groups, updating the state of lights, as well as changing the scenes. The library is tested, fully working, asynchronous and includes error handling.

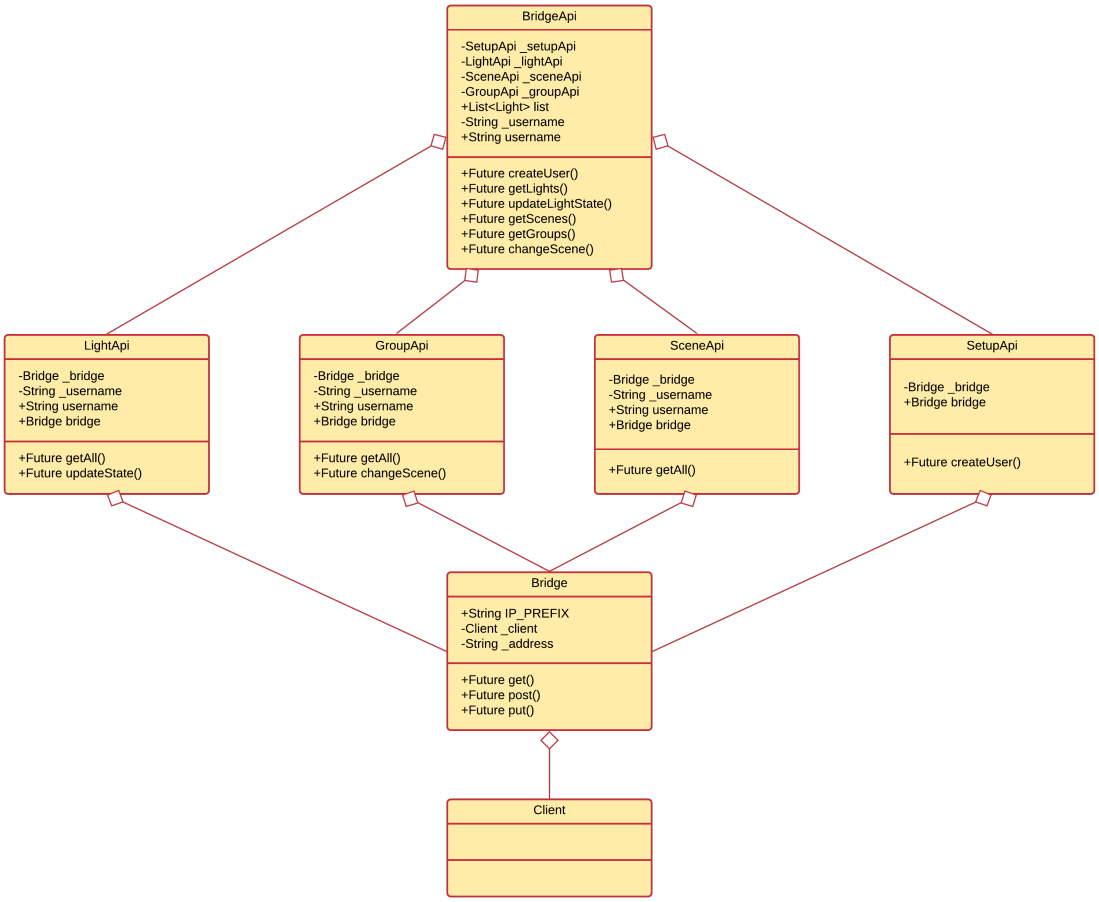


Figure 31: Library class diagram

The source code for the Philips Hue Flutter Library can be found in appendix C.

4.5.1 Class: HueApi

The HueApi class is the only part of our application that communicates directly with the library. The API-calls and information is handled in this class. This class is a Singleton, which is explained in 2.5.1. This is to make sure that there is only one instance at any given time. After the HueApi is initialized, it runs the setup method. This method applies the library to look for a bridge on the same network. The method connects to the first one it finds. When it is connected, the rest of the calls are made available. Some examples of available functionality is to turn lights on or off, or change the scene.

4.5.2 Class: BridgeApi

When an application uses the library, the only API class needed is the BridgeApi class. A measure made for other developers who wants to utilize this library, was to let the BridgeApi-class handle all of the API calls to other classes, as well as initialize them. This makes sure the developer only have to keep track of - and initialize one class. To make it easier for the developer that wants to use the library it initializes and handles every call to all the other API classes, so that the developer only needs to keep track of and initialize one class, this can be seen in listing 10.

Listing 10: Simple methods in the BridgeApi-class.

```
class BridgeApi {
  final SetupApi _setupApi;
  final LightApi _lightApi;
  final SceneApi _sceneApi;
  final GroupApi _groupApi;

  ...

  Future<List<Light>> getLights() async {
    return await _lightApi.getAll();
  }

  Future<List<Scene>> getScenes() async {
    return await _sceneApi.getAll();
  }

  Future<List<Group>> getGroups() async {
    return await _groupApi.getAll();
  }

  Future<void> changeScene(String sceneId, groupId) async {
    return await _groupApi.changeScene(sceneId, groupId);
  }
```

```
  Future<void> updateLightState(String id, LightState state) async {
    return await _lightApi.updateState(id, state);
  }
}
```

4.5.3 Classes: LightApi, SceneApi, GroupApi, SetupApi

Instead of having all the calls directly in the HueApi class, it was created a hierarchy. The HueApi-class calls on the BridgeApi-class, which then calls on four different classes. The first class is the *LightApi*-class. This class handles API calls related to the lights and the "light state". It can update the state of lights, turning them on/off, and change color and brightness. The second class is the *SceneApi*-class. SceneApi handles API calls related to the scenes. The third class is the *GroupApi*. This class handles the functionality to change scenes, meaning they change the state of all lights in the group. The reason that GroupApi has this feature and not SceneApi is how the Philips Hue team have written their API. The user change the scene of the group instead of updating an already existing scene. Lastly, SetupApi is the class that handles user creation on the bridge. These classes then use the Bridge class to do REST API calls to the physical Philips Hue Bridge. The responses are then parsed and returned to the HueApi as objects. All API classes have functionality to get information about their respective objects.

4.5.4 Class: Bridge

The Bridge class does the REST API GET, POST and PUT requests to the physical Philips Hue Bridge. This class is used by the other API classes to convert the request into JSON format and convert the JSON response back into objects. It also keeps track of the IP address, which removes the necessity of sending it with every call.

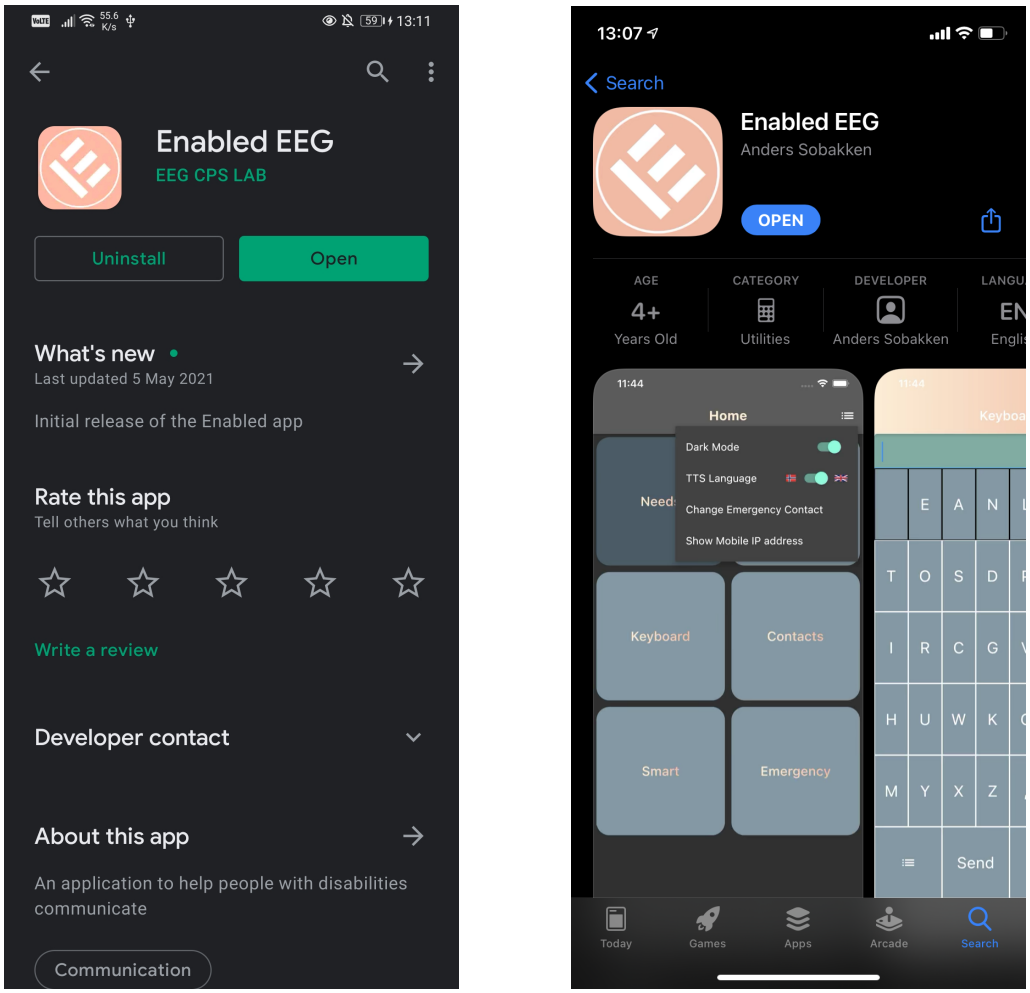
4.6 Connection Between Mobile and Desktop Application

One of the main objectives of this project were to develop a desktop application that connects the Emotiv headset to the mobile application. The desktop application connects

to the headset and forwards the data to the mobile application. The mobile and desktop application connects through a socket connection. The mobile application hosts a WebSocket server and is waiting for devices to connect, while the desktop application creates a WebSocket client and connects to the server through the IP address of the phone. A WebSocket need a Domain Name to establish a secure connection, and because the application is using an IP address to connect to the server, the socket connection is unsecured. The client sends the command received from the headset to the server. This object contains simple text-strings and no sensitive data. The command object can either be a Facial Expression command or a Mental Command. The server receives and interprets the objects and translates them to the commands Left, Right, Select and Back. These commands lets the user move around in the application and select different functionality, with the use of their brain activity.

4.7 Releases

One of the main objectives of the group was to create a mobile application that was available for both iOS and Android devices. As a result, the mobile application were released on the Google Play Store and the Apple App Store.



(a) A picture of the app in the Google Play Store (b) A picture of the app in the App Store

Figure 32: Pictures of the application on the distribution platforms

Another main objective was to create a desktop application that runs on both the Windows and macOS platform. The group created an application on both platforms. Since desktop application does not have their own release platform, such as mobile applications, it was decided to create a simple website, where the user would be able to download the application.

4.8 Website

To keep all of the information about the desktop and mobile application on the same place, and to have an easy way to promote our app, the group created a simple website. Just like the desktop app, the website is created using React. However, as this is not supposed to be a desktop application there was no need to use Electron.

4.8.1 Pages

The website has four pages, these are the Front page, About us, Download, and a last page for the privacy policy. Navigation between the pages are managed by the navigation bar in the header. We wanted these pages to resemble the mobile application in design, therefore it uses the same color plate.

Front Page

The main page includes a short description about the mobile application and the desktop application and the use of it. On the page is also a demo video showing the functionality of the applications and how to setup the mobile application.

About Us

The about us page goes into more details about the project, the project group, and each member of the group. Here you are also able to find links to every GitHub page for each member and the GitHub link for the projects. In addition to this there will be a link to the bachelor thesis for further information.

Download

The download page includes all the download links for both the desktop application and the mobile application.

Privacy Policy

App Store demands a privacy policy, which is a policy about how data is collected from the application and then used, to release the mobile application. This page contains the privacy notice and all of the information about how user data is collected, stored and handled.

5 Discussion

In this chapter we will discuss the results we have accomplished and delve into why we have ended on the results we produced. We will also discuss how this result deviated from what we set out to get. Lastly, we discuss what we have learned from this project and what we can take with us from this project into future work and projects.

5.1 Comparison to the Original Issue

The original objective of the project did not deviate far from the achieved result. The initial aim for the group was to have an app ready for release on iOS and Android. The app managed to be ready for release on the App store for iOS and the Play store for Android. We also finished the desktop app, which is now available on all platforms. We have implemented text-to-speech, several preset messages used for communication, the ability to make custom messages and a keyboard with similar layout to the Accessible Virtual Keyboard bachelor thesis. We have also implemented the smart equipment, Philips hue, into our application. Chromecast implementation was also attempted, but was never fulfilled due to the complexity, time constraints, and lack of support in Flutter. The supervisors suggested implementation of *Steady State Visually Evoked Potential* into our application, but it was postponed because of the time constraint and complexity of the task.

5.2 Quality of Work

The project goal was to improve communication, independence, and overall life quality for the person with ALS, or other physical disabilities. It was important to implement features that could assist a wide spectrum of disabilities, to improve the quality of life of as many people as possible.

Desktop App

To make the Enabled app available to as many people as possible we created a desktop app with ease of use in mind. The desktop app works on macOS, linux, and windows to make it available to as many users as possible. The desktop is simplistic in style and interaction, to make the setup as easy as possible for people with little computer experience.

Mobile App

To accommodate for the different symptoms someone with ALS may experience several steps were made in the Enabled app. We implemented text-to-speech in several of the pages of the app, like the needs-page, custom-page, and keyboard-page. Text-to-speech assists those who have lost the ability to speak with communication with those around them. Losing the ability to speak is also a symptom almost every patient with ALS is going to experience. We also wanted to accommodate for those who still are able to speak. To do this we implemented an instant call feature in one of our pages, this allows the person to call directly from the Enabled App. This allows people who are wheelchair bound or unable to move their body, but still are able to talk to make phone calls to anyone.

5.3 Choice of technology

When we started planning we realized there were a lot of different languages, SDKs, and frameworks that could be used for what we wanted. Here we discuss why we chose the technologies we did, and our experiences with them.

5.3.1 Flutter

The reason the group chose flutter is that one of the goals was to develop a mobile application to be able to work for both iOS and Android users. Some of the group members have previously worked with Android Studio and Xcode. The experiences from working with these platforms was that it took a while to develop apps. Subsequently, the group figured out that we would spend a lot of our time if we were to develop both apps on their native platforms separately.

Previously, we talked with fellow students who recommended Flutter for app development. They described Flutter as a platform that was easy to learn, and at the same time it beats the other development platforms in results. This is because developers are able to write one code for two platforms. Based on these recommendations we decided on Flutter for our project.

The choice of Flutter proved to be an efficient and solid solution for our project. The SDK exceeded all of our expectations and the platform proved to be a great platform for

developing mobile apps. Even though Flutter is a relatively new SDK, it offers great documentation for every aspect of app development, whether it is creating custom widgets, tests, or releases for Play Store and App Store. The language used, Dart, is pretty similar to languages the group was familiar with, like Java and C#. This gave us an easier time learning Dart than a language with a different syntax. However, how Flutter widgets were created was new to us, but it was more confusing than difficult, and the group quickly grasped the concept.

The biggest issue we encountered was navigation using a bottom navigation bar as well as button presses. A mobile application is usually developed for a user that can press on the screen, and not to be controlled by only four inputs. Being restricted to only four inputs meant that we always had to keep track of what button was hovered to simulate a button press if the Push command was sent for example. All in all, the group is very pleased with Flutter and Dart as tools to create the mobile application.

5.3.2 React

The reason we chose to develop the desktop application in React was that we wanted to learn it and it is similar to Flutter in many ways. React is also a known framework, and it seemed like the better option compared to regular HTML, CSS and TypeScript.

The use and learning of React was something the group experienced positively. React has a clear and comprehensive official documentation with API references, and coverage of concepts and best practices. This made it quick to start development, with a knowledge-based starting point. React also has a large and active ecosystem, which made it easier to fix bugs, as well as finding resources. However, it was still a completely new framework for the group members, which resulted in some challenges.

One of our biggest challenges in the beginning was the React-classes and the React functional components. There are two different ways to create components. We started out the project by creating different class-components. We chose the class-components since it seemed similar to the objective programming we were used to. As we progressed, we learned that React recommended the use of functional components. We had to adjust to this and refactor the components. The functional components was harder to compre-

hend in the beginning, but as we learned how to use them it was easier than the class-components.

Another challenge that arose was the useState functionality. The group members were used to be able to set the values of variables directly without minding mutation, which means alter the variable. One of the most prominent features of React is the ability to quickly discern what piece of data has changed and only update the affected component. This took some time for the group members to get used to, but was a great tool when we wrapped our head around the concept.

5.3.3 Electron

We were considering several options before we chose the Electron framework for developing the desktop application. When we first started creating an assessment of the project, we wanted to continue on the already implemented desktop application. We downloaded the application and tested it. The first thing we noticed was that the application crashed. This was most likely because of a *NullPointerException*. The second thing we noticed was that the application used the computers command-line to show information about the process. We talked to one of the developers and he said it was "a hassle" to use the GUI framework he used to develop the application, since a lot of the functionality he wanted to use was deprecated or had little to no documentation.

After we had written off the last application as an option, we began to look into different options to develop the GUI. We looked into *.NET* GUI frameworks, but figured that we were only able to create an application for the Windows platform. We then moved away from this option, since we would like to create an application for all platforms. The second option we considered was Flutter. Flutter had functionality to create desktop applications, but this was only in alpha at the time. The thought process behind choosing flutter was to keep the project consistent. If we were to develop the mobile and desktop application on Flutter, it would make it easier for a group to inherit the project and keep working on it. After considering it for a while, we refrained from using it since it was still in alpha-stage.

In the end we settled on Electron js. The group decided to use a boilerplate project for Electron. This boilerplate implements React into the development. Two weeks after

the group decided to use Electron, Google released *Flutter 2*, which had better desktop support. Even though the Flutter desktop had moved from alpha to beta, we still decided to stick with Electron because of the limited time we had to create a product.

We were grateful that we chose a boilerplate for Electron. Before deciding to integrate the boilerplate into the project, we tried to configure the system our self. This proved to be quite complex and time consuming, since none of the group members had worked with something similar before. Even though applying the boilerplate was time efficient, it also brought some problems along. By using this boilerplate, we had accumulated a lot of source code and the complexity of the project had grown quite a bit. The boilerplate also integrated external scripts written in languages such as ruby, which made it even more comprehensive.

5.3.4 Typescript

There were several reasons why we chose Typescript over JavaScript. The first reason why we chose Typescript was that we wanted to limit the type-error at run-time. Typescript introduces an optional strong static typing, which means once a variable is declared it can not change the type. This makes the code more predictable, which enhances the likelihood of a function working as intended. The second reason we chose Typescript was because we wanted learn the language. Several students had recommended it to us and explained that they felt that Typescript gave their code more structure and readability compared to JavaScript.

The steep learning curve of Typescript was a challenge as the group members had close to no experience with JavaScript or Typescript before commencing on this project. This made it especially hard to understand Typescript, since most of the examples online are written in JavaScript. This resulted in a rough start before we learned the syntax. To choose Typescript over JavaScript was an advantage for the group. This was especially true for the use of types and interfaces

Even though most of the code in the desktop application is written in React. We deliberately chose to write the Emotiv driver class in Typescript. The reason being that we wanted the driver class to be available independent of what framework the developers

chose.

5.3.5 Material UI

The group chose to complement our react code with the Material UI framework from Google. The group found several different design frameworks, but went with Material UI because it used the *Material design*. This is the same design as the one we are using in our mobile app. Another benefit of choosing Material UI was that we did not have to spend as much time on designing components.

Choosing Material UI as a component library was a great choice. It reduced the time the group spent on designing components. Since the Material UI component are developed and maintained by a professional company, it is safe to assume they worked as intended.

5.4 Testing

Testing is a subject the group had little to no experience with, and the little we had was only connected to unit testing. This section describes the experiences with unit and component testing in both the mobile and desktop application.

5.4.1 Mobile Testing

The component testing of the mobile application was implemented to ensure correct and efficient behaviour of components. This was done by following general principles in component testing as described in 2.7.3. In general, the group is satisfied with the tests we have created. They have allowed us to focus on more important features instead of debugging and manual testing. This was possible because the tests showed if the intended behaviour was broken or not. Another positive element of testing was that fewer bugs managed to get into production builds without being detected. As a result, we were able to release better versions of the application.

The group did not have any experience with component testing and only minimal experience with unit testing before starting on this project. As a result, we faced some issues in the beginning. The first problem was how the components were set up inside the tests.

The components could not just be created, but had to be wrapped in a MaterialApp first. A second issue was that it took a while to understand what was returned from the methods that supposedly found widgets. The assumption was that they returned the specific widget, but instead they returned on object of type Finder containing information about the widget. Luckily Flutter had great documentation on the subject and we quickly learned the correct way of writing component tests.

The tests were valuable tools when implementing new features. We were able to run the tests in a matter of seconds to see if everything still worked as intended. Once the tests were implemented they worked for the remainder of the project.

5.4.2 Desktop Testing

As for the desktop testing, the group decided to implement component testing. We decided that it was important to ensure that the components behaved - and displayed correctly. The group is overall pleased with the component testing of the desktop application. The tests are simple, well documented, and follows the Material UI guidelines. A module the group did not get around to test was the Emotiv Driver. This class communicates with the Emotiv API and the headset through a WebSocket connection. The module relies on asynchronous requests and the response depends on the Emotiv API. Each response returns a promise object. When the group researched how to implement tests for a socket client, there was little to be found. Some of the solutions the group found was to mock a socket server. This turned out to be cumbersome. Firstly, the group did not create the WebSocket server. Therefore, we had problems mocking the server. Secondly, the documentation of the API was lacking. The group encountered several responses from the API that was not documented. That being the case, we found it difficult to test if the socket client behaved correctly.

The group understands that unit testing for the Emotiv driver would benefit the product. The driver class is one of our most important classes in the desktop application. It would be reassuring to have tests to rely on, to ensure that everything works. However, this was prolonged because of lacking documentation, the complexity and the time limit set by the project. As a result, the group decided to focus on finishing other important tasks instead. Something the group discussed was that if we were to create something like this again,

we would write tests for components and classes earlier in the development.

5.4.3 Usability Testing

The group wanted to perform usability testing on the mobile application to see if the design and layout was user friendly. The test included three different tasks that would test the most frequently used features in the app. The features tested would have been navigating, adding new elements, typing, and calling.

When the test regime was planned the corona situation in Ålesund was more stable and the rules were not as strict, but this later changed when we wanted to carry out the tests. Using the Nielsen Normans Groups research [45], we found out that we wanted to have around six users testing to maximize the result. As a result of the new rules it was not possible to get as many testers as we would like. This was because the people we were in close contact to had either seen the app before and would not give the objective outcome the test wanted, or it would not be allowed according to the rules.

5.5 Comparison to Existing Technology

When we started this project the first thing we did was to look at some technologies that were already out there to get some inspiration for our project. The research gave us an overall view about the restrictions for our project, what we could expect to get done and where we could see the technology move towards the next few years. We found that the field of work had just been started.

5.5.1 Accessible Virtual Keyboard 2017 Bachelor Thesis

The first things the group realised when testing the old application from the 2017 bachelor, was that it did not have a way to communicate to the user. There was no feedback when the input was registered or sent, we wanted to improve on this by making the feedback better and implement more functionality. This would help fulfill the main goal of the project, which was to improve communication, independence, and overall life quality for the person with ALS.

Enabled EEG offers more features then Accessible Virtual Keyboard(AVK). The first feature is text-to-speech. Enabled EEG makes it possible for the user to communicate with their friends and family by the help of sound. The text-to-speech feature is available in English and Norwegian. Whereas, AVK relied on the device screen to communicate with others. This makes it easier for the user to communicate, without having to depend on both people seeing the screen. Another feature Enabled EEG offers is the Philip HUE smart solutions. The user is offered more independence and are able to control the lights at their home. Enabled EEG is also available on more platforms. The user is able to utilize the application on iOS and Android devices. Whereas, AVK is only available for Android devices.

The possibility of making calls is another feature that the Enabled EEG application offers, that was not available in AVK. This feature enables users with ALS that still can talk, or people with other diseases that restricts them from moving, to call friends and family without assistance, making them more independent. AVK were thinking about making their keyboards able to send text messages, however this is not a feature that was added.

Another advantage with Enabled EEG is that it make use of four different commands from the Emotiv headset. Four inputs makes it easier for the user to navigate the application and the keyboard. AVK utilizes two inputs. As a result, an error with the navigation is more noticeable, since the user needs to navigate further to go back to the starting point.

The advantage of the 2017 Bachelor was that it had more keyboard layouts. These layouts were optimized for specific uses and could therefore be used for different purposes. The Chess keyboard from this bachelor could be a good addition to our app. However, the implementation was deemed redundant without a chess-board built into the application, since the user is able to play through the normal keyboard.

5.5.2 Previous Desktop Application

There are several differences between the new and the previous desktop application The first difference is that the new application is available on more platforms. It can run on Windows, mac OS and Linux. In comparison to the previous application that was only available on the Windows platform.

The new application has better error handling. If a user tries to connect to a socket server on an IP address where there is no server running, the new application will alert the user that an error occurred. Furthermore, the application will tell the user the most likely reason for the error. Whereas, the previous application crashes.

This brings us to the third point. The new application has better feedback. The previous application relied on the terminal window to give feedback to the user. The target group of this application is users with reduced functional capacity and it is not fair to assume that they are all familiar with the terminal window. Furthermore, the feedback messages consisted of short messages, which were hard to understand for a non-developer. On the other hand, the new application gives the user feedback in the user interface. The feedback mostly relies on alerts. The feedback messages have been made more generic, which makes it easier for the user to understand what is wrong, and how to correct the error.

The new application offers more functionality for the users. The previous application gave the users the option to add their IP-address, select their profile and connect to the Web-Socket server. The new application offers the same functionality, as well as an additional stream and the functionality of being able to set their headset sensitivity.

The new application provides more information to the user. The previous application relied on external text documents to show the users how to use the application and how to find the IP address of their. The new application has this integrated into the user interface.

5.5.3 SimpliHere

SimpliHere gave us a lot of inspiration on how to set up the app, but we wanted to make the app in a way where the client could be more independent when used. SimpliHere uses a voice assistant to navigate and help the user in the app. We thought this would be hard for the user to do by themselves, and it would be difficult for the user in the later stages of the disease. Using EEG headsets instead allows people with ALS to benefit from the application in both early and later stages of the disease, giving them a consistent tool throughout.

5.5.4 Next Mind

Since the Next Mind headsets uses an overlay from Unity to create applications it does not have the same scalability as our application. It is restricted to what is in the Next Mind SDK and what features Unity offers.

One advantage of using the Next Mind headset is that the setup is much simpler compared to the Emotiv headset we used. The Next mind headset can be put on and start instantly, while the Emotiv headset needs to be hydrated with saline and fitted more precisely so the sensors get a good connection.

5.5.5 Neuralink

The EEG headsets used in our project use 14 electrodes to measure brain activity and convert to simple outputs like Push, Pull, Left, and Right. In comparison, Neuralink uses 1500 electrodes giving it a lot more potential and precision than what is currently available. Our application was restricted to only four inputs, because the Emotiv application only allows a maximum of four commands active at the same time. With Neuralink having more than 100 times as many electrodes, the possibility for more than four inputs is great, which would make it much easier to navigate and use features in the Enabled application.

A disadvantage of the Neuralink is that it is an implant. Compared to relatively cheap headsets that can just be set up and used by anyone. The implant requires surgery to use, making it unavailable for most people and also very expensive. However, as this completely new technology it is safe to assume that the continuous development will make it more accessible at a later stage.

5.6 Further Work on the Project

There are some things that could be worked on both when it comes to the mobile application, and also in the desktop application.

5.6.1 Mobile Application

One of the main upgrades which could be done with the mobile application is give it more functionality. The functionality the group believe would do the most for the users are smart solutions like *Chromecast/Apple TV, Smart Thermostats* with more. The group think there will be a huge spike in usages of smart equipment technologies in the coming years. Implementing several smart solutions into the application is a good way of giving users more individuality and control.

Another implementation that our application may have a need for, is a way to send text messages directly through the app. This would enable the user to keep in touch with other people or communicate urgent needs. It would also be a great addition to the contact page, where you are able to call the person. The message page would also help out those who might not have the strongest vocal cords or have a paralysis, that might hinder their speaking abilities.

Optimizing and making the dictionary better could be a future implementation. This could make it for the user to send messages as well as the general use of the keyboard. The current dictionary works fine, but it would be a good way to make the application feel smoother.

5.6.2 Desktop Application

There are several ways to improve to the desktop application. One of them is to add the training of a profile. This is something the Emotiv API supports, but since the user has to use the EmotivBCI application when using the desktop and sending information, it was deemed redundant. Another function the Emotiv API provides is the ability to return feedback on how well the command was executed. By showing the feedback, the user is able to see if they need to train their commands further.

The Emotiv team is currently working on getting the EmotivBCI and Emotiv App as a mobile application. If this application turns out successful, this could be a good substitute for the desktop app as you would not have to have a secondary app running in the background, since the mobile would be able to do it all. This would make the application more user friendly since it would be easier to set up and have both applications running

on the phone.

5.7 Experiences from the Project

As the projects comes to an end, it is important to reflect on the project. This section will the discuss the experiences the group members had with development methodologies, different technologies and time planning. It will also address what it was like to work in a group and how we solved problems that arose and the experiences the group members can carry over to new projects.

5.7.1 Use of Development Methodology

For this project we chose Scrum as our development methodology. When the group started working on the project we did not know every feature that needed to be implemented in the applications, or which features depended on the others. The flexibility of Scrum allowed the group to plan and update new features as we needed them. The Scrum sprints lasted two weeks, and started and ended on Mondays. We had a weekly standup meeting on Mondays, to make sure we were on track with our tasks. Another benefit of having the sprints on Mondays were that it gave us good time to prepare for the meeting with our supervisors on Wednesday. This made sure we always had something that the supervisors could give input on.

The group had little to no familiarity with Scrum before this project. We quickly realized that, by applying Scrum to the project in a backlog, it allowed us to have a better control of our time management. The sprint meetings could be used to update our Gantt chart and add new tasks to the backlog. This was able to be done without interfering with the entire project structure. This is something that would have been a risk using the Waterfall method. This resulted in positive experiences using Scrum.

For our project we chose to use two week long sprints, with sprint review meetings after each sprint. The reason we chose this sprint format, was because we feared that having one week sprints would not give us enough time to implement features that could not be split into smaller parts. On the other hand, four week sprints would not let us update the project often enough. If we were to discover new, needed features during a sprint, we

would have to wait a long time before we could add it to our scope. This could cause us to not have enough time to implement the new feature.

At the start of the project we also considered using the Extreme programming methodology, but we quickly decided to just implement a couple practices from the methodology instead. Pair programming and collective code ownership are two of the practices we included from Extreme programming. When the desktop application development process started, the group split into two pairs, where one pair worked on the mobile app and the other on the desktop app. As we worked in these pairs, usually only one from each pair actually wrote code, while the other reviewed and gave feedback. We had great experiences with this and it increased the quality of our code. Everyone in the group had access to review and update all of the code in the full system, including mobile and desktop app, as well as the website. Collective code ownership like this made sure that the entire team was familiar with the full system and helped us not write redundant code.

5.7.2 Use of Backlog / Jira

We used a backlog at Jira to manage our tasks and sprints. We mostly used the backlog to divide main task into smaller tasks and split them between the group members. This way we assured that every member contributed close to equally each sprint. One thing we could improve on to the next project is to move a task from in progress to done. Aside from that, it was a great tool to plan and manage our project.

5.7.3 Use of Wireframes

The group decided to create wireframes of the mobile application before we started programming. The wireframes were useful tools to our development. The first benefit of creating wireframes was that the group could create a visual representation of the layout without using a lot of time on programming. The second benefit was that the group could use the wireframes when we began programming. The group noticed it was easier to create something when you already knew the layout of what you are creating.

5.7.4 Use of Git and GitHub

Git is a wonderful tool which enables us to keep version control of our code, as well as improve the teamwork and structure of the project. In this project it was essential for our progress and productivity.

The group encountered several problems while developing which was solved by Git. One of the most prominent problems was when the desktop application stopped working on a later stage of the project. We started debugging and were not able to find what caused the error. With the help of Git we were able to roll back to a previous version of application that worked and continue our work from there.

One thing we noticed a bit to late was the auto-generated files from Xcode when group members were working from a mac. Two of our members worked from a mac, and since we had different developer teams on Xcode, a lot of the files changed for each commit. This required us to do some extra configurations each time the mac users had to merge code. This could be solved by buying a developer team from apple, but we refrained from it because of the price.

5.7.5 Time Management

When we started the project, we tried to account for the time each main task would take and wrote it down in the pre-project report. We then created a Gantt chart from these estimations, which can be found in appendix . We have used the Gantt chart throughout the project to estimate if we were on schedule. We also used the Gantt chart to make changes and estimate if we had time to do the task or not. The Gantt chart has been a great tool to estimate and plan the project.

5.7.6 Communication

As described in the pre-project report, the group were to have weekly meetings each Monday. This was to discuss the progress of the project, obstacles that occurred, and unforeseen events that would change the course of the project. Another meeting that was planned was the daily standup meeting. This meeting were used to talk about the plan

for the day. The group complied with the scheduled meetings throughout the entirety of the project. This resulted in productive work days as well as a proper structure in the everyday life, which improved communication within the group, since we were able to talk about and discuss the project every workday. Because of the pandemic that we are currently in, most of the communication were digital. This meant mostly working from our apartments. We solved this issue by creating a *Discord server*, which was used to communicate and share resources between the group members.

5.7.7 Working as a Team

During this project we had a lot of focus on using teamwork as a tool to improve our workflow and quality. In the beginning of the project we started of as one team to get started and get to know flutter. Later in the project we decided to split into two teams of two, where one team was going to focus on the desktop application and the other team was going to finish the mobile application. Doing it like this we could use pair programming to ensure quality in our product.

6 Conclusion

The overall purpose of the project was to give users with the disease ALS, as well as people with other disabilities, an opportunity to communicate with others and give them more independence using smart solutions. This was solved by using a headset from Emotiv which read the brainwave activity.

This project offered opportunities for the group to immerse ourselves with new technologies and frameworks. The focus throughout the bachelor thesis has been on creating user-friendly application as well as applying design- and programming principles to our project. Our project resulted in a desktop application that is available on both Windows and macOS, as well as a mobile application that can be used by both android and iOS devices. The desktop application connects the headset from Emotiv to the mobile application. Using these applications, the user has the ability to communicate, call, and control smart lights. This means that people with disabilities are given the opportunity to communicate, as well as increase their independence, using less complex equipment. The group also developed a Philips Hue lighting library and a website where the user is able to download the applications. We also see a development in the smart equipment sector. Smart solutions are being developed more frequently and many of these have open APIs, which makes it possible to link the equipment to our mobile application.

The bachelor project has been very educational. The group learned more about how to organize a larger project and what tools we can use to increase the chances of the project being successful. One of the major success factors in this project was how we planned the project from the start. We spent a lot of time estimating how long each of the major tasks would take and then we did our best to meet the time allotted. It also helped that we used scrum as a development methodology. Scrum is flexible, which enabled us to make reservations about unforeseen events. All in all, it has been a demanding process, with great learning benefits, both in technology and in the development of a product. The final product includes features which meet initial requirements and objectives, while it still has room for improvement.

Bibliography

[1] Daily Stand-ups vs. Weekly Updates, May 2018. URL https://getweeklyupdate.com/blog/daily-stand-ups-vs-weekly-updates/. Section: Communication.

[2] A. V. K. . EEG Accessible Virtual Keyboard, 2017. URL https://github.com/accessible-virtual-keyboard.

[3] AltexSoft. Extreme Programming: Values, Principles, and Practices, Jan. 2021. URL https://www.altexsoft.com/blog/business/extreme-programming-values-principles-and-practices/.

[4] A. Altvater. What is Agile Methodology? Tools, Best Practices & More, Sept. 2017. URL https://stackify.com/agile-methodology/.

[5] ArpitAsati. What is web socket and how it is different from the HTTP?, Dec. 2019. URL https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/. Section: GBlog.

[6] Atlassian. A brief overview of Confluence, 2021. URL https://www.atlassian.com/software/confluence/guides/get-started/confluence-overview.

[7] Atlassian. Jira | Issue & Project Tracking Software, 2021. URL https://www.atlassian.com/software/jira.

[8] A. Betts. Building Hybrid Applications with Electron, Oct. 2016. URL https://slack.engineering/building-hybrid-applications-with-electron/. Section: Uncategorized.

[9] K. Bradshaw. [Update: Flutter confirmed] Google Stadia app is now live in the Play Store, Nov. 2019. URL https://9to5google.com/2019/11/08/google-stadia-app-play-store-download/.

[10] B. Chaperon. Cortex API, 2021. URL https://emotiv.gitbook.io/cortex-api/.

[11] CloudFare. What is the Internet Protocol? | Cloudflare, 2021. URL https://www.cloudflare.com/en-gb/learning/network-layer/internet-protocol/.

[12] Dart. Dart overview, 2021. URL https://dart.dev/overview.html.

[13] Dexxor. Git, May 2021. URL https://en.wikipedia.org/w/index.php?title=Git&oldid=1021092067. Page Version ID: 1021092067.

[14] M. Dryka and O. Gierszal. What Is Electron.js? Features, Architecture and Tools, Apr. 2021. URL https://brainhub.eu/library/what-is-electron-js/.

[15] EMOTIV. About Emotiv, 2021. URL https://www.emotiv.com/about-emotiv/.

[16] Emotiv. Emotiv BCI - Built for Insight and Epoc+ headsets, 2021. URL https://www.emotiv.com/emotiv-bci/.

[17] EMOTIV. What is an EEG Headset? Definition & FAQs, 2021. URL https://www.emotiv.com/glossary/eeg-headset/.

[18] P. Engineering. Node.js at PayPal, July 2018. URL https://medium.com/paypal-tech/node-js-at-paypal-4e2d1d08ce4f.

[19] Enginess. The 6 Principles Of Design, a la Donald Norman, Mar. 2014. URL https://www.enginess.io/insights/6-principles-design-la-donald-norman.

[20] Fenix5567. Neuralink - Wikipedia, July 2021. URL https://en.wikipedia.org/wiki/Neuralink.

[21] M. Fowler. Software Architecture Guide, Jan. 2019. URL https://martinfowler.com/architecture/.

[22] D. Gerard. GitHub, May 2021. URL https://en.wikipedia.org/w/index.php?title=GitHub&oldid=1021173718. Page Version ID: 1021173718.

[23] Graham87. Electroencephalography, Apr. 2021. URL https://en.wikipedia.org/w/index.php?title=Electroencephalography&oldid=1019570663. Page Version ID: 1019570663.

[24] E. Gregersen. HTML | Definition & Facts, Oct. 2020. URL https://www.britannica.com/technology/HTML.

[25] B. Group. The My BMW app: new features and tech insights for March 2021., Mar. 2021. URL https://www.press.bmwgroup.com/global/article/detail/T0328610EN/the-my-bmw-app:-new-features-and-tech-insights-for-march-2021?language=en.

[26] R. Guru. Facade, 2021. URL https://refactoring.guru/design-patterns/facade.

[27] Guru99. What is Component Testing? Techniques, Example Test Cases, Mar. 2021. URL https://www.guru99.com/component-testing.html.

[28] hoangphamemotiv. GitHub - Emotiv/cortex-v2-example: Example with Cortex V2 API, Apr. 2021. URL https://github.com/Emotiv/cortex-v2-example.

[29] HSS. What Causes ALS, Who Gets It and What Are the Symptoms ?, 2021. URL https://www.hss.edu/condition-list_amyotrophic-lateral-sclerosis.asp.

[30] F. Inc. React – A JavaScript library for building user interfaces, 2021. URL https://reactjs.org/.

[31] P. M. Institute. What is Project Management?, 2021. URL https://www.pmi.org/about/learn-about-pmi/what-is-project-management.

[32] IONOS. Observer pattern: what does the observer design pattern do?, Oct. 2020. URL https://www.ionos.com/digitalguide/websites/web-development/what-is-the-observer-pattern/.

[33] joaquinelio. An Introduction to JavaScript, Apr. 2021. URL https://javascript.info/intro.

[34] D. Johnson. What is Chromium? A guide to Google's open-source software project, which runs some of the world's most popular internet browsers, Nov. 2020. URL https://www.businessinsider.com/what-is-chromium.

[35] M. Julian. Getting Started in Lucidchart, May 2017. URL /blog/getting-started-in-lucidchart.

[36] Kate Moran. Usability Testing 101, Jan. 2019. URL https://www.nngroup.com/articles/usability-testing-101/.

[37] R. Kostrzewski. 10 Famous Apps Using ReactJS Nowadays, Aug. 2021. URL https://brainhub.eu/library/famous-apps-using-reactjs/.

[38] Masem. Discord (software), Apr. 2021. URL https://en.wikipedia.org/w/index.php?title=Discord_(software)&oldid=1019800674. Page Version ID: 1019800674.

[39] Material-UI. Material-UI: A popular React UI framework, 2021. URL https://material-ui.com/.

[40] MDN. Learn to style HTML using CSS - Learn web development | MDN, Apr. 2021. URL https://developer.mozilla.org/en-US/docs/Learn/CSS.

[41] Microsoft. Typed JavaScript at Any Scale., 2021. URL https://www.typescriptlang.org/.

[42] I. Montiel. Low Coupling, High Cohesion, Sept. 2018. URL https://medium.com/clarityhub/low-coupling-high-cohesion-3610e35ac4a6.

[43] S. M. Nes. En kort introduksjon til Scrum, May 2019. URL https://www.visma.no/blogg/en-kort-introduksjon-til-scrum/.

[44] NextMind. Technology, 2021. URL https://www.next-mind.com/technology/.

[45] J. Nielsen. Why You Only Need to Test with 5 Users, Mar. 2000. URL https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

[46] E. E. Nilsen, arildhv, and R. T. Bye. NTNU-IE-IIR/excited-eeg, Nov. 2020. URL https://github.com/NTNU-IE-IIR/excited-eeg. original-date: 2020-06-23T20:13:40Z.

[47] Node.js. Introduction to Node.js, 2021. URL https://nodejs.dev/learn. Section: Node.js.

[48] C. Ovesen, K. S. V. Jacobsen, and K. A. Olsen. EEG Controlled Robot Arm. May 2019. URL https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2610894. Accepted: 2019-08-25T14:09:01Z Publisher: NTNU.

[49] S. Padmanabhan. How We Built eBay's First Node.js Application, May 2013. URL https://tech.ebayinc.com/engineering/how-we-built-ebays-first-node-js-application/.

[50] U. Pisuwala. A comprehensive guide on agile methods for modern software development, Oct. 2018. URL https://www.peerbits.com/blog/agile-software-development.html. Section: Software.

[51] H. Reactor. What is JavaScript Used For?, Oct. 2018. URL https://www.hackreactor.com/blog/what-is-javascript-used-for.

[52] E. Rodriguez. electroencephalography | Definition, Procedure, & Uses, Oct. 2017. URL https://www.britannica.com/science/electroencephalography.

[53] A. Rolstadås. forprosjekt, May 2018. URL http://snl.no/forprosjekt.

[54] J. J. Shih, D. J. Krusienski, and J. R. Wolpaw. Brain-Computer Interfaces in Medicine. *Mayo Clinic Proceedings*, 87(3):268–279, Mar. 2012. ISSN 0025-6196. doi: 10.1016/j.mayocp.2011.12.008. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3497935/.

[55] SimpliHere. SimpliHere - SimpliHere - Simplifying caregiving for those impacted by ALS and other neurodegenerative diseases., 2021. URL https://simplihere.com/.

[56] C. Software. What is Flutter? Here is everything you should know, Aug. 2019. URL https://medium.com/@concisesoftware/what-is-flutter-here-is-everything-you-should-know-faed3836253f.

[57] SourceMaking. Design Patterns and Refactoring, 2021. URL https://sourcemaking.com.

[58] C. Sprague and L. McKenzie. eBay Motors: Accelerating With Flutter™, Sept. 2020. URL https://tech.ebayinc.com/product/ebay-motors-accelerating-with-fluttertm/.

[59] A. Staff. Microsoft's new Code editor is built on Google's Chromium, Apr. 2015. URL https://arstechnica.com/information-technology/2015/04/microsofts-new-code-editor-is-built-on-googles-chromium/.

[60] Study. Pros & Cons of Agile Development Methods - Business Class [2021 Video], 2021. URL https://study.com/academy/lesson/pros-cons-of-agile-development-methods.html.

[61] SYNOPSYS. What Is Software Architecture & Software Security Design and How Does It Work? | Synopsys, 2021. URL https://www.synopsys.com/glossary/what-is-software-architecture.html.

[62] L. Tebartz van Elst, M. Fleck, S. Bartels, D.-M. Altenmüller, A. Riedel, E. Bubl, S. Matthies, B. Feige, E. Perlov, and D. Endres. Increased Prevalence of Intermittent Rhythmic Delta or Theta Activity (IRDA/IRTA) in the Electroencephalograms (EEGs) of Patients with Borderline Personality Disorder. *Frontiers in Behavioral Neuroscience*, 10, Feb. 2016. doi: 10.3389/fnbeh.2016.00012.

[63] N. Technology. Making Netflix.com Faster, Apr. 2017. URL https://netflixtechblog.com/making-netflix-com-faster-f95d15f2e972.

[64] A. Tikkanen. Tim Berners-Lee | Biography, Education, Internet, Contributions, & Facts, Feb. 2021. URL https://www.britannica.com/biography/Tim-Berners-Lee.

[65] Tutorialspoint. TypeScript - Overview - Tutorialspoint, 2021. URL https://www.tutorialspoint.com/typescript/typescript_overview.htm.

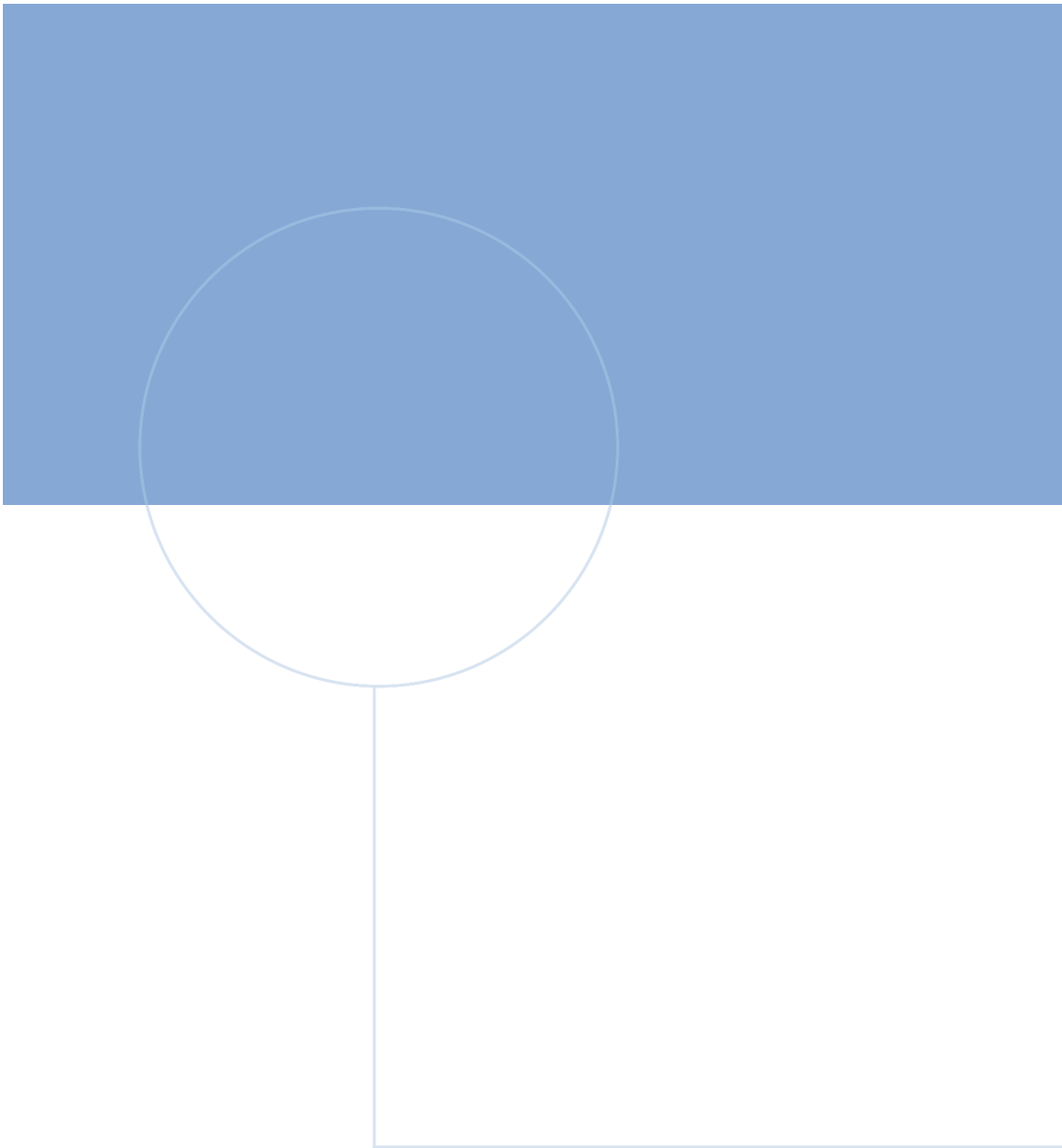
[66] O. Universitetssykehus. Hva er ALS?, June 2017. URL https://www.helsenorge.no/sykdom/hjerne-og-nerver/als/.

[67] E. Vedes. How to develop your React superpowers with the Container Pattern, Oct. 2018. URL https://www.freecodecamp.org/news/react-superpowers-container-pattern-20d664bdae65/.

[68] T. Verplaetse, F. Sanfilippo, A. Rutle, O. Osen, and R. T. Bye. *On Usage Of EEG Brain Control For Rehabilitation Of Stroke Patients*. ECMS European Council for Modelling and Simulation, 2016. ISBN 978-0-9932440-2-5. doi: 10.7148/2016-0544. URL https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2498928. Accepted: 2018-05-23T12:06:22Z Publication Title: 544-553.

[69] E. İNAN. What is Unit Testing?, July 2020. URL https://medium.com/hardwareandro/what-is-unit-testing-435e8134b16e.

Appendix



NTNU

Norwegian University of
Science and Technology