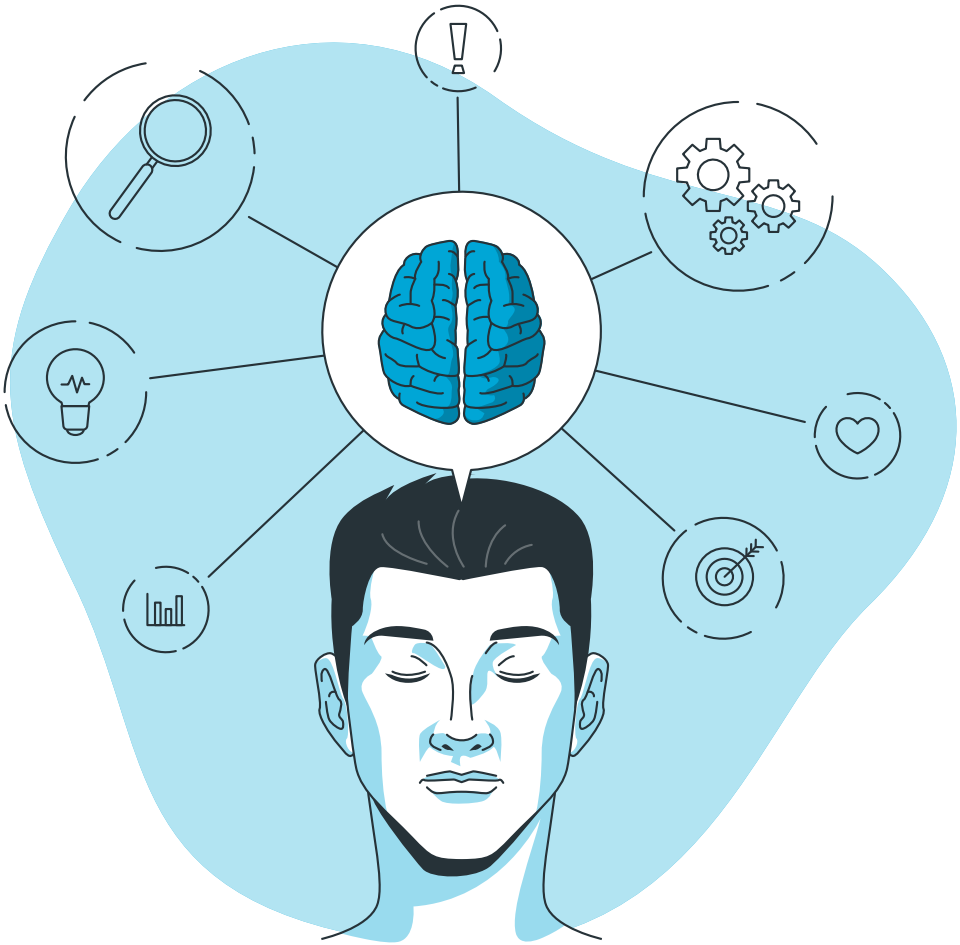
EEG-Based Brain Computer Interface

Interface: Real-time Control

Tijn Bakkum
Davi Spiller Beltrao



EEG-Based Brain Computer Interface

Interface: Real-time Control

by

Tijn Bakkum
Davi Spiller Beltrao

to obtain the degree of Bachelor of Science in Electrical Engineering
at the Delft University of Technology.

Students:	Tijn Bakkum (5498414)
	Davi Spiller Beltrao (5465265)
Supervisor:	Dr. B. Hunyadi
Thesis Committee:	Dr.ir. R.F. Remis
	Dr. B. Hunyadi
	Dr. C Gao

Cover:	'Mind map Cartoon Illustrations' by Storyset, free for personal and commercial purpose with attribution.
Style:	TU Delft Report-Thesis Template by Daan Zwaneveld (with additional modifications by Pragun Srivastava).

TU Delft

Abstract

In this thesis, the development of a graphical user interface for a brain computer interface (BCI) system is discussed. This system is based on electroencephalographic (EEG) signals from motor imagery (MI). BCI applications for low consumer-grade are very limited, since most of the applications are towards medical use or games. This limits the potential BCI can have, since it only applies to one specific environment. The goal of this thesis is to show that a low consumer-grade BCI can be made, ranging from educational to personal use, with the needed quality and usability. This is achieved by showing the possibility of distinguishing MI signals. Specifically in this thesis, left hand, right hand, tongue and feet. This can be done and utilized through a machine learning algorithm. Presentation methods for this are displaying EEG signals and using demos that show the purpose of the BCI system. An example of such a demo is moving a simulated computer cursor. In order to have this system for a wider public, the interface should be able to adapt to each individual user using personalized machine learning models. Due to the distinct, personal EEG patterns, the accuracy of classifying the MI-EEG signals using personalized ML models is much higher than with an ML model trained solely on a public dataset. For more advanced personal use of the system, the interface contains everything needed regarding EEG signals, including real-time data streaming in order to directly see change in EEG signals and to help the user detect potential errors in the measurement setup.

Preface

This thesis 'EEG-Based Brain Computer Interface: Interface: Real-time Control' has been written for the Bachelor Graduation Project and to obtain the Bachelors degree of Electrical Engineering at the Delft University of Technology. This project was commissioned by dr. B. Hunyadi with the main goal to develop a BCI using brain waves and distinguish motor imagery signals from test subjects. Those signals can be converted to digital outputs to control for example a computer cursor. There were six people working on the project, who were subdivided into three subgroups: preprocessig, decoding and time-control and interface. This thesis is focused on the time-control and interface part of the project.

It was really thrilling to work on a project that is on the horizon of science and the latest technology and we think the future for BCIs can be very bright. We would like to thank our supervisor dr. B. Hunyadi for guiding us through this BAP. We would also like to give our thanks to prof.dr.ir L. Abelmann for giving advice during the project. And finally, we would like to thank our fellow team members, Jack Chen, Martin Little, Victor van der Doorn and Pragun Srivastava, for all their hard work over the last two months.

Tijn Bakkum
Davi Spiller Beltrao
Delft, June 2024

Contents

1 Introduction

1.1. Background information

To understand the goal of this thesis, it is important to understand what a Brain Computer Interface, henceforth referred to as BCI, exactly is. A BCI is a system that enables a connection between the human brain and a computer system without the need of any muscular action [24]. The human brain activity is usually obtained from measured electroencephalogram (EEG) signals. These measurements are, most of the times, acquired by applying noninvasive electrodes on the user's scalp [2], in the form of a 'cap'. In the case of this thesis, the *Unicorn Hybrid Black* by g.tec was used [8]. It will be referred to as the EEG cap. These EEG signals are then classified by a machine learning model, which results in an action in the interface.

BCIs have a large scale of potential use cases and are currently already put to use in many fields. Examples include: Medical applications, controlling smart environments, educational purposes, applications in the entertainment industry and more [1].

1.2. The goal of this project

During this project, the goal was to create a program that can be used to demonstrate the possibility of distinguishing motor imagery (MI) signals. Specifically right and left hand, tongue and feet. To achieve this, the project was divided into three separate subgroups; The preprocessing subgroup, the decoding subgroup and the interface subgroup. The preprocessing subgroup was responsible for recording and filtering the incoming data from the EEG cap. The decoding subgroup was responsible for training a model on EEG data that could decode the data received from the preprocessing subgroup. Lastly, the interface group, which will be the focus of this thesis, was responsible for creating a graphical user interface (GUI). All this together had to create an overall product to stream data from the EEG cap and turn the decoded EEG signals into visual actions, such as moving a computer cursor. An overview of the system can be seen in Figure 1.1.

1.3. Problem definition

Even though BCIs have been an upcoming technology for at least three decades now with research papers dating back to 1990 [31], most BCIs are still not very accessible for users without any knowledge of EEG data, and are mostly focused on medical use. It's very difficult for these users to utilize and understand these interfaces. In the last decade there have been papers researching the use of BCI in combination with the entertainment industry, in particular with the gaming industry [29]. These applications are already focused on consumers instead of medical applications. However, these are still very specialized in the area they are invested in. Firstly, the BCI only works for the specific medical application/game. There is no flexibility in usage. Secondly, for a good experience, the system has to be trained to the specific user due to each user's specific neural patterns. These are not yet offered in these industries. As Vasiljevi mentions, there are very few studies that analyze the usability and qualitative conditions of the user's interaction with the BCI [29]. These are mitigated by applying gel to the EEG cap for example, which is really uncomfortable.
Therefore the focus of this project will be making an interface for a non professional consumer targeted BCI. This means that the BCI system is supposed to work for multiple users, can be used for educational or personal uses, showing the data from the EEG cap with demos for showing the possible purposes of BCI. Granting so the consumer the possibility to operate it in diverse applications, with the necessary usability and quality, without the discomfort.

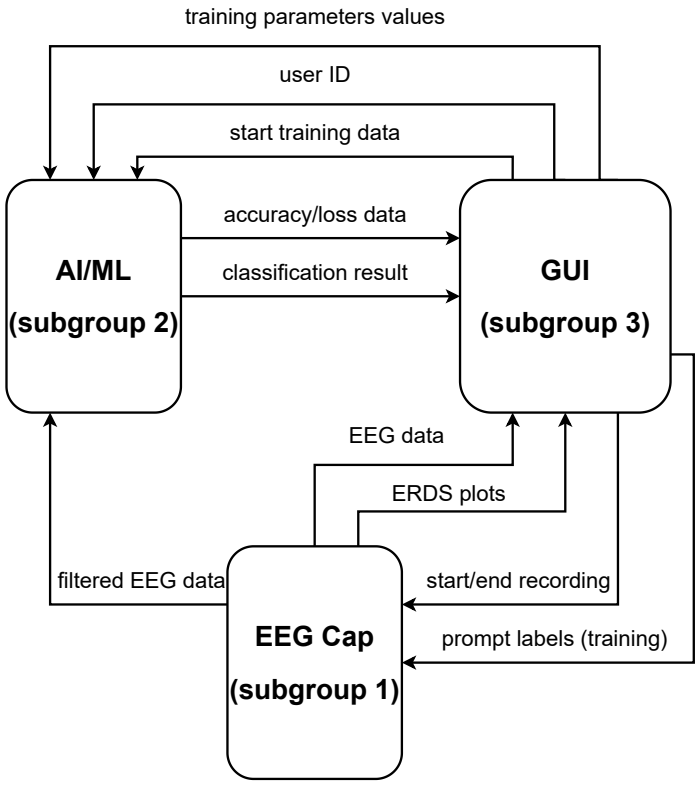


Figure 1.1: Overview of the whole system, divided into three subgroups, with the signals that are sent between them.

1.4. Structure of this thesis

This thesis will be divided in multiple chapters corresponding with different segments of the project. In the following chapter, chapter 2, the program of requirements for this project will be presented. In chapter 3, the personalization of the interface according to the distinct EEG-signals of users will be examined. In chapter 4, the process of bringing real-time data streaming to life will be shown. Thereafter, the process of visually representing the decoded signals will be laid out in chapter 5. In chapter 6 the integration with the other two modules, the preprocessing and the decoding groups, will be discussed. Finally, the resulting interface will be constructed from the building blocks discussed in the previous chapters, in chapter 7. The thesis will be concluded in chapter 8, where future work and points of improvement will be discussed and the conclusion of this thesis. In the appendices the source code can be found for the interface (chapter 9), the scheduled planning through the BAP period (chapter 10) and screenshots of the resulting GUI (chapter 11).

2 Program of Requirements

In order to evaluate the success of the project, requirements are made. Starting from the overall requirement of the whole system, narrowing it down to more specific requirements, focused on this subgroup.

2.1. General requirements

As mentioned in the Introduction, the goal of this project is to make a system that demonstrates the possibility of distinguishing motor imagery (MI) signals using the g.tex Unicorn Hybrid Black. After doing research and receiving specific requirements from the costumer, the following requirements were set:

2.1.1. Functional requirements

A1. The system must be able to decode MI-EEG signals to determine the intended movement, in accordance with requirements B1 - B3.

A2. The system must be able to visualize the decoded MI-EEG signals through interactive demonstration(s).

A3. The system must be able to plot EEG data in real-time (note: we define a real-time plot as one which satisfies requirement B4).

A4. The data handling performed by the system must be individualized to each user.

2.1.2. Performance requirements

B1. The minimal sensitivity of prediction for each of the 4 motor imagery actions separately must exceed 50%.

B2. The average overall accuracy of prediction across all 4 motor imagery actions must exceed 70%.

B3. The action latency must be less than 5 seconds.

B4. The plot latency must be less than 1 second.

As Morash [20] discusses, there are differences in MI signals between limbs. One evident indication, is the fact people are right handed or left handed. Therefore it was decided to make requirements for the distinct accuracy between MI signals and the overall accuracy of prediction.

2.1.3. Implementation requirements

C1. The data utilized by the system must be measured using the g.tec Unicorn Hybrid Black.

C2. All software developed for the system must be written in Python. This includes software for measurement, decoding, and visualization.

We define the action latency as the time between the moment at which the prompt is shown on the GUI, and the moment at which the corresponding classification is shown.
We define the plot latency as the time between the measurement of a sample and its corresponding output to the plot.

2.2. Interface requirements

After discussing with the client and the other subgroups and having the general requirements and goals in thought, the following requirements were set for the interface subgroup:

2.2.1. Preprocessing for interface

- The data streaming functionality from the preprocessing subgroup must be implemented into the interface (only visually).
- The data recording and filtering from the preprocessing group must be integrated into the subsystem.

These requirements follow from the goal of this project. In order to make a system that demonstrates the possibility of distinguishing MI signals using the g.tex Unicorn Hybrid Black, the data collected from it has to be shown on the interface and be communicated to the decoding part. What is involved is the ability to handle the data in such a way that the user can rapidly recognize the different MI signals or even correct errors during recording and the demos. These will also reflect on the performance requirement of the plot latency. Reminder that the requirements of the quality of the measured signal are outside the scope of this subgroup.

2.2.2. Decoding for interface

- The decoding subsystem must be integrated into the subsystem.
- The interface needs to show relevant data of the decoding subsystem in the form of an accuracy and a loss plot.

From the decoding subgroup and the goal of this project these requirements were set. The different MI signals come to life when they are classified into computer actions. Using the features, the classification can be tested. In order to train the data properly, it was asked to add the latter two plots. These will also reflect on the performance requirement of the action delay.

2.2.3. Interface functionality

- The interface must be functional independent of input from other modules.

Meaning that the interface should behave equally when integrated and not integrated with the other subgroups. For example, which later in this report will be explained more: the interface should be able to plot, independent of the input (e.g. cap or random). The mouse cursor should move. Beginning with the keys WASD for example.

- The interface must show related plots to the data; an FFT, band power and ERDS plot.
- The user must be able to switch through the 8 channels for the FFT and band power plots.

When doing research, it was clear that the useful information from EEG signals were frequency based. Depending on the MI actions, the amplitude and power within certain frequencies increase or decrease, hence showing these plots will help distinguishing the MI signals. Further explanation will be done in chapter 4.

- The user needs to be able to start and stop the recording from the interface.
- The user needs to be able to change values relevant to the training cycle.

In order to train the machine learning model, data has to be recorded and certain training values need to be changed. This will help increasing the average overall accuracy.

- The experiment setup must be implemented into the interface in a separate window.

This requirement was given by the client. This way, it will give flexibility to the user and it can guaranty consistency in recording and during demos. It can also make it possible to have a user with the cap and another person controlling the interface, for example with two screens. This makes sure the user sits still, for better EEG measurements.

3 Individualized Data Handling

3.1. Individual differences

As discussed by Lopez [16] and Melnik [19], it is important to realize the difference in EEG signals between users. Since MI asks a lot cognitively, the differences should not be undermined: "However, ratio of variation over subjects was smallest in the ''low-level'' tasks like flickering checkerboard and highest in the more cognitive-oriented tasks like decision-making task" [16]. This difference has also a lot of influence on the accuracy of the ML model [25]. In order to improve the possibility of distinguishing MI signals and the accuracy of classifying them, the machine learning model has to be trained user-distinct. When training the model to a specific user, it will be easier to distinguish the signals for that specific user and having the option to change the user in the whole system, will automatically improve the system in general. This can be implemented by adding a window in the GUI, where the user can be selected. Additionally a recording protocol should be made, to record and ultimately train the ML model specifically to the chosen user. But more can be done than only training the model. The user itself can also be trained.

3.2. User trainability

As said previously, MI systems require a lot of cognitive effort, because the user has to actively think about a certain movement. There are other ways of measuring EEG signals. For example using evoked signals, flashing light for example. For evoked systems, you passively wait for something to be 'evoked' and then the user reacts to it. Hence systems that require the user to be active, like MI systems, are called active systems and systems where the user waits for a trigger are called reactive systems. "The biggest disadvantage of active systems is the need for extensive user-specific training sessions"[21]. So it's not only important to be user-specific, but there are also lots of benefits to train the user itself and not just the model.
As Kayagil et al.[12] describes, there are simple things a user can do to eliminate artifacts from the measured data. Examples are not blinking, moving the eyes or any other muscle. The lesser the user does these during recording, the better the signal will look like. More advanced ways are through neurofeedback. Neurofeedback is a form of feedback where the EEG signals are measured and translated into visual or feedback[18]. Using the example of the cursor, when controlling it, it gives visual feedback to the user and gives them the possibility to calibrate/adapt to the sensitivity of the system [22]. Specifically, how well does the system respond to the type of movement the user is thinking. Is there a certain movement that the system doesn't respond too or is there a movement that is easier to think about that has the same result. This way the user can adapt, due to the neurofeedback it receives from the interface, improving the possibility of classifying the intended movement correctly.

3.3. Interface

In summary, the system should be individualized, meaning that the classification system adapts to a specific user. Additionally, a training phase should be added to improve the overall accuracy of the system. This can be done through passive measures and neurofeedback. Passive, by giving prompts of what part of the body the user should think of and neurofeedback through interactive demos.
Overall, making the system handle data of a specific user will improve the possibility of distinguishing MI signals and the overall accuracy of prediction.

4 Real-time Data Streaming

The data quality of the BCI might vary depending on the location and the setup of the experiment. This means that before actually using the BCI for recording data, the user should be able to look at the incoming data from the EEG cap. This way the user can see if there are any errors during setup or strong interferences, and thus prevent any faulty measurements. This chapter will go into the design choices that make this data streaming module work.

4.1. The Headset

The use of an EEG cap was explained in the introduction, in particular the use of the *Unicorn Hybrid Black* by g.tec that can be seen in Figure 4.1. This headset is used to measure the user's brainwaves using EEG signals. These signals are retrieved by electrodes placed on the user's scalp with the use of the EEG cap. In total, eight channels are received from the EEG cap with a sample frequency of 250 Hz.

Figure 4.1: The Unicorn Hybrid Black from g.tec [8]

4.2. Visualization Methods

If the user needs to be able to detect potential errors during the setup, it is important to know what type of data visualizations should be used to guide the user. Because the only data that is available is the data from the eight channels of the EEG cap, there is a limited amount of possible visualizations. Considered visualizations of the EEG data were:

- The eight channels of EEG data.
- Fast Fourier Transform (FFT) of each channel.
- Band powers of each channel.
- Scalp voltage topographies.
- Evoked response desynchronisation and synchronisation (ERDS) plots.

4.2.1. Eight Channel EEG data

The data received from the EEG cap consists of data from the eight electrodes placed withing the cap on the skull. The locations for these electrodes are shown in Figure 4.2.

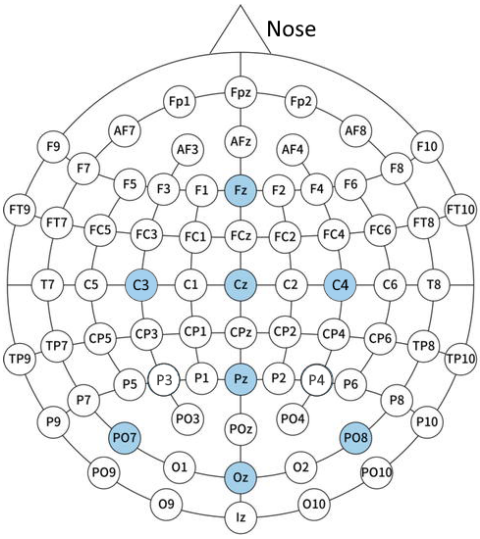


Figure 4.2: The positions of the electrodes (indicated in blue). Modified image from [27]

These eight channels are important to be shown to the user, as they are the data directly from the EEG cap. This means that in case of an error, such as an electrode not being connected correctly, it will be easily detectable using these eight channel plots.

4.2.2. FFT

The Fast Fourier Transform (FFT) is a fast and, more importantly, a more efficient way of calculating a Discrete Fourier Transform (DFT) [5]. The DFT itself is a form of the Fourier Transform, but using discrete time samples instead of continuous time samples. The goal of these transform functions is to transform the data from time domain to frequency domain.
Showing EEG data in frequency domain instead of time domain can be useful. Brainwaves can show rhythmic behaviour at times. An example of this, is when someone closes their eyes, there is a high possibility of rhythmic patterns appearing with a frequency between 8 and 13 Hertz as can be seen in Figure 4.3 [7]. This is called an alpha wave.

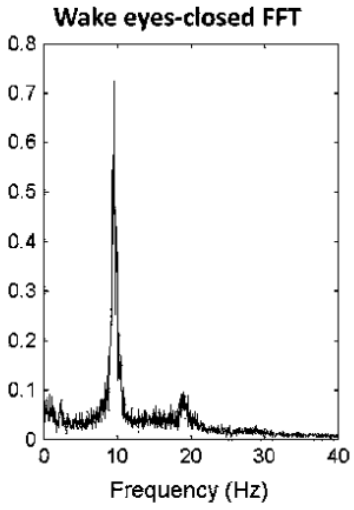


Figure 4.3: FFT of Eyes Closed EEG signal from [28]

The EEG frequency spectrum is often divided in five frequency bands: Alpha, Beta, Delta, Theta and Gamma waves. Their respective frequencies can be seen in Table 4.1. And because these are all frequency related, FFTs can be used to easily show if there is a spike somewhere in those ranges, just like the eyes-closed FFT in Figure 4.3 showing a peak on 10 Hz. Additionally, the location in the brain is also important. For example, the most active part of the brain is the opposite side of the (intended) movement. So if the user thinks of moving the right hand, the left side of the brain will be more active. Another difference is the function of a location in the brain. The frontal part of the brain after the fissura longitudinales (middle of the brain) is responsible for motor activity for example, while the back is more active when relaxed/sleeping.

Terminology of bands	**Frequency band [Hz]**	**Position**
Delta	< 4	F3, F4, Fz
Theta	4 to 8	F3, F4, T3, T4, Fz
Alpha	8 to 13	O1, O2, P3, P4, Pz
Mu	8 to 13	C3, C4, Cz
Beta	13 to 30	F3, F4, C3, C4, Cz
Gamma	30 to 100	P3, P4, T3, T4, F3, F4

Table 4.1: EEG Frequency bands [7] and the relevant electrodes where they are most active [15].

The FFT or DFT equation is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \tag{4.1}$$

$x[n]$ represents a time signal of length N here. k ranges from 0 to $N-1$. $X[k]$ is a discrete signal representing the DFT of the input signal $x[n]$. The sum is taken over all the samples n which range from 0 to $N-1$. Using Equation 4.1 the FFT will be calculated on the EEG signal.

By using an FFT, the user can see whether these frequency bands are active, and thus see whether the system works. If, for example, there is no spike visible in the alpha frequency band while the user is closing their eyes or the left electrodes are not active when thinking of right hand, there is probably something wrong with the setup. Additionally, certain EEG-MI signals can also be detected, depending on the activity of the bands for specific electrodes. As can be seen in Table 4.1, alpha and mu have the same band frequency, but the positions are very different. Mu is more active for MI, more center of the brain, and alpha more the back. Not only does the position of the electrode give insight, but also the bands itself. Studies show there are differences between them. For example, mu and beta are most relevant when detecting MI of the hands [32].
Thus, it is important to show the FFT plot of the channels.

4.2.3. Band power

As was just discussed, EEG signals exist of multiple waves. The band power refers to the power present in a specific frequency band. In this case the power per frequency band as seen in Table 4.1.

The power P_{band} in a specific frequency band of an EEG signal is calculated using the following equation:

$$P_{\text{band}} = \sum_{k=k_1}^{k_2} |X[k]|^2 \tag{4.2}$$

Here k1 and k2 are the frequency boundaries of the chosen frequency band. $X[k]$ is the calculated FFT as explained in subsection 4.2.2. Using Equation 4.2 the band powers will be calculated for the discussed frequency bands.

Looking at power present in a certain frequency band can give more insight than just looking at the FFT plot. Taking, again, the example of closing your eyes, the power inside the alpha frequency band will be significantly higher than when your eyes are opened [14].
The same points discussed previously can be applied to the band power plot. However, there are advantages. Firstly, band power is more simplified and highlights the frequency bands in 4.1, making it easier to interpret the data, especially for users less familiar with signal processing. Secondly, having the predefined frequency bands, there is potential to directly relate the bands to their functional states as explained in subsection 4.2.2. Lastly, the power within a band can be averaged, helping reduce the impact of noise and artifacts that might affect the FFT plot.

4.2.4. Scalp voltage topographies

Scalp voltage topographies show the EEG voltages on the scalp. Different voltage levels are displayed using different colors. An example of scalp voltage topographies can be seen in Figure 4.4.

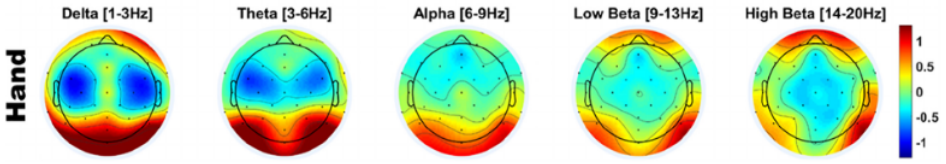


Figure 4.4: Example of Scalp Voltage Topography from [10]

In this figure, the topography is divided in frequency bands instead of in channels, but it still shows the function of the scalp voltage topography: the user can see what part of the brain is active at what point. However, this does not add something directly when having the eight channels, FFT and band power plots. It does give more clarity to the relation between activity and position on the scalp, but it would be too much work for little additional insight. Therefore it was decided to not include this type of EEG data visualization.

4.2.5. ERDS plots

Different stimuli around a person can create a difference in evoked responses in their brain waves with a predictable delay. ERDS plots are used to visualize these brain wave responses. ERDS is divided into two parts: Event-related Desynchronization (ERD) and Event-related Synchronization (ERS). ERD is the decrease in power in a specific frequency band, while ERS is the increase in power in a specific frequency band. Taking the eye closing example again, this evokes an increase in power in the alpha band and is thus an example of ERS.

ERDS plots show how the EEG signal changes in response to an event. The data for these plots are usually computed using the following steps [23]:

1. Using a bandpass filter on the data of one particular event.
2. Calculate the power by squaring the amplitude of the data.
3. Average the power across the data of the event.
4. Average the data over time.

The fact that the ERDS plots need the entire data of an event, indicates that this method is not suited for live visualization of EEG data. Thus it was decided to not include this in the live visualization part of the project.
However ERDS plots still can be useful to detect when recorded data might be faulty. For example, when not seeing an increase in the power in the alpha frequency band during a period of rest, there is probably an error in the setup. That is why it was decided to still make the ERDS plots available after recording the data of the user.

4.2.6. conclusion

The final choices for visualizing the EEG data were showing the eight EEG channels, the fast fourier transform and the power in the frequency bands of these channels, and for after recording, the ERDS plots.

4.3. Real-time plotting

In order to detect any potential errors using the visualization of the EEG data, it is important that the data is shown with a delay that is negligible. To recapitulate, in the requirements it was stated that the plot latency must be less than one second. This plot latency is defined as the time between the measurement of a sample in the EEG cap and its corresponding output being shown in a plot. Requirements for this are:

- Retrieving data from the EEG cap.
- Ability to plot the data.
- Minimizing the plot latency.

4.3.1. Retrieving data from the EEG cap

In the design of the GUI, it is important to take into account how the preprocessing and measurement subgroup has implemented the connection with the EEG cap. The implementation is briefly explained in this part.

Communication with the EEG cap is achieved using the Lab Streaming Layer (LSL). This is a software that enables the collection of data from hardware, such as the EEG cap, and is able to collect raw data from the EEG cap with a frequency of 250 Hz. To retrieve the data from the LSL software, the pyLSL library is used [26]. When connected to the EEG cap, samples can be pulled using the library. By having a continuous loop active, every cycle a sample is pulled. One sample exists of 17 values: The eight EEG channels, the three-dimension gyroscope and accelerometer of the EEG cap, an internal counter, validation indicator and battery life. For visualizing the EEG data, only the eight EEG channels of this list are needed. Each time a sample is taken, the EEG data is stored in an array which keeps track of 50 samples. The size of this array was tested to make sure there was not too much data present in the plots, increasing the plot latency, and not too less, giving no insight on what is going on. Whenever a new sample is pulled, the first sample in the array gets deleted, decreasing the plot latency.

4.3.2. Plotting the data

When plotting the data, the allowed plot latency is the main focus. This means that the library used for plotting the data should be very lightweight. Two considerations for plotting libraries are:

- Matplotlib
- PyQtGraph

Matplotlib

The first possible library is Matplotlib. Matplotlib is one of the biggest libraries for plotting data. This makes it easy to find advice for potential difficulties and most people have experience with it. However, a big disadvantage of Matplotlib is that it is not optimized for high frequency live plotting, which is needed to prevent a plot latency above one second.

PyQtGraph

Another possible library for plotting the EEG data is PyQtGraph. It is a library built on top of PyQt, made for creating interfaces in python. This means that if this library is to be used, for the entire GUI PyQt has to be used, which is no problem, as the entire GUI will be built using PyQt as will be discussed in chapter 7. A benefit of PyQtGraph is that it is much faster than Matplotlib. While it is much faster, it does lack in features as opposed to Matplotlib. Luckily, for live plotting data, such features are not needed.

After analyzing these advantages and disadvantages, PyQtGraph became the library used for plotting the data.

As explained in subsection 4.3.1, every update a new sample gets pulled. This new sample gets added to the data array which is then plotted onto the eight plots, one plot for each EEG channel. In case of the FFT plot and band power plots, the actual plotting has is also done in real-time. The only difference is that all points are sampled at the same time, no sliding window. The case of the ERDS plots, as mentioned before, is slightly different. The data is calculated and plotted by the measurement and preprocessing subgroup. The plots are received and shown on the interface.

4.3.3. Optimizing plot latency

Even with the use of PyQt, there are still some plot latency problems. PyLSL, the library used for collecting samples from the EEG cap, actually holds samples in a buffer. So the longer the plot update takes, the more samples are stored in the buffer due to the time needed to update the data array and update the plots. Consequently, the plotting will start to increasingly fall behind due to the buffer.

To solve this, the samples are pulled as fast as possible. But while the samples are pulled, only every fiftieth sample is actually put inside the array and used for updating the plots. This way only every 50 samples, the calculations and the plotting of the data have to be executed, resulting in no significant delay. Only using every fiftieth sample, and throwing away the other samples is called downsampling. This downsampling is only for visualizing the EEG data for the eight channels and will therefore not result in aliasing in the plots, as the preprocessing group will use all the samples available.

4.4. Other plots

After recording the data, the user will be able to start the machine learning training process on the newly acquired data. At that time, there is no need to show the live FFT and band power plots. However, there is some data from the training process that can be useful to be shown. So when the recording is done and the training can begin, the FFT and band power plots will be turned into an accuracy plot and a loss plot respectively. Each iteration of the training process, the accuracy and loss are calculated and sent to the GUI. Accuracy and loss are important data to analyse the performance of the newly trained model. When the accuracy is very low or the loss is extremely high, there is a possibility something is wrong with the data, and therefore it is a good indication of when a user has to record new data or not.

4.5. Conclusion

The plots visualized in the interface are the eight EEG channels, FFT, band power, ERDS, accuracy and loss. In order to facilitate integration with the other subgroups, the backend code has to be implemented. Meaning that the interface should plot independently of the other subgroups, but in order to save time, the code and interface should be ready for any input of the other subgroups. This integration is describe in chapter 6.

5 Visual Representation of the Decoded Signals

An important requirement for the interface, is the ability to visualize the decoded signals from the decoding group. For visualizing the data, there are a few options to consider:

- Simulating Computer Cursor Movement.
- Controlling a Wheelchair.
- Games.
- Controlling a character in virtual reality.

Simulating a computer cursor movement is a very famous demo when it comes to BCI systems. The only part that has to be implemented is moving an object in four directions (or even better with click system).
Controlling a wheelchair is already being researched [21]. However, this costs too much time to actually implement safely while also working on the other parts of the GUI.
Playing games with a BCI system is also being researched, as was previously discussed in chapter 1. The freedom of choosing the difficulty makes this a perfect option to visualize the decoded MI signals.
Lastly controlling a character in virtual reality using decoded MI signals sounds really promising. This however would be difficult to combine with wearing an EEG cap, and thus is not a great option.

The final choices to visualize the decoded signals were simulating computer cursor movement and playing games. For this, a module for controlling a simulated computer cursor and two games are made. In this chapter the design choices and the code of the demos will be discussed.

5.1. Cursor control

The cursor moves in four different direction, each MI signal representing one direction. This was implemented completely in PyQt, the library used for creating the GUI, more about this will be discussed in chapter 7. The cursor is a moving image controlled by the decoded signals. As one of the requirements stated, the interface must be functional independent from input from other modules. This means the cursor movement code works with keyboard control, but can easily be integrated with the decoded signals.

With the simulated computer cursor implemented, the window can be seen in Figure 5.1.

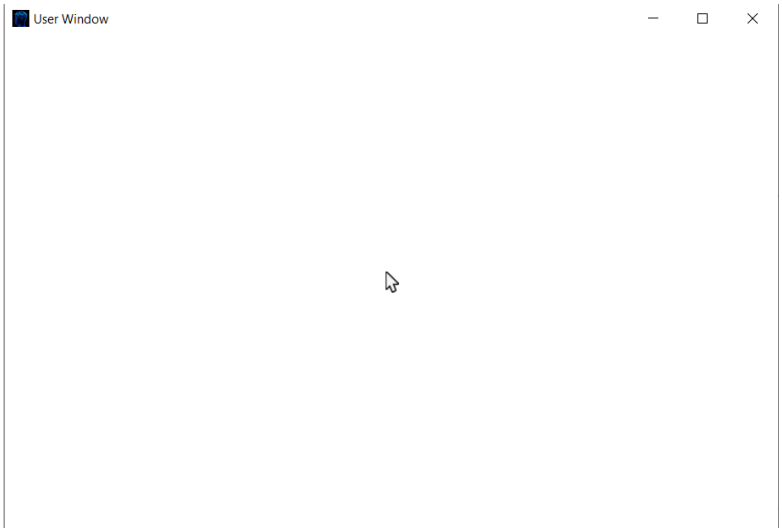


Figure 5.1: Simulated Computer Cursor

5.2. Games

Two simple games were developed for the user to interact with the BCI system, using only PyQt to avoid the difficulties of integrating extra libraries into the GUI. This approach also adds an element of fun, enhancing the user's engagement with the system.

An important consideration is the action latency from decoding the EEG signals when actually using the BCI. While the cursor movement is generally unaffected by this delay, the games are affected. When it takes a maximum of five seconds before an action is performed, it can cause problems when the game changes in that time. Therefore, the games were designed to give the user enough time to react and dodge obstacles effectively.

The two games developed are *The Floor is Lava* and *Meteor Shooter*, which will be explained in the following sections.

5.2.1. The floor is lava

Floor is lava is a game where the user has to avoid the tiles that are going to be lava (red). By moving the player (blue square) the user can avoid the red tiles. Before a tile turns red, it first turns to yellow to warn the user that in a few moments, that tile won't be accessible. First, the game is made to move with the WASD keys, but implementing with the other subgroups gives the user the possibility to move it with their mind. The controls will be: left and right will be left and right hand respectively, up is using the tongue and down using the feet.

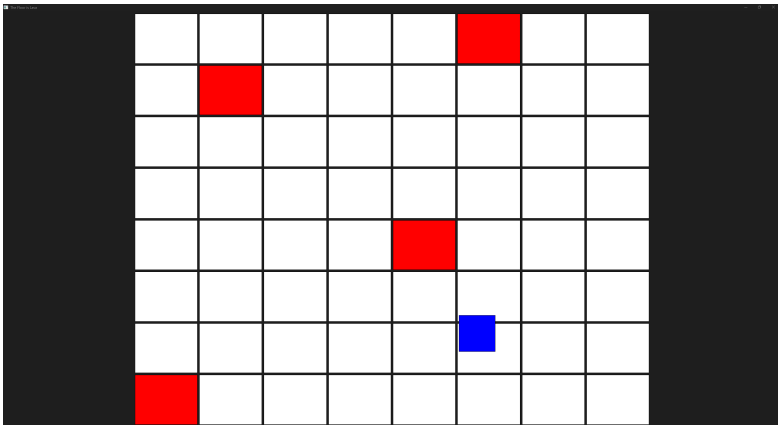


Figure 5.2: The floor is lava

5.2.2. Meteor Shooter

Meteor shooter is (very) loosely based on Space Invaders, a very classic arcade game from 1978. In Space invaders you shoot at aliens that are slowly closing in on you. In meteor shooter, the player has to shoot on incoming meteors, when using your imagination. In Figure 5.3, the game can be seen. The player (the green rectangle) can move left and right. This will be controlled by the BCI using the left hand and right hand MI, but again, to satisfy the requirement of the interface having to be functional without any input form other modules, the input was simulated using the A and D keys, for moving left and right respectively. The player can also shoot using the space key. When actually implementing the decoded signal, Left and right will be controlled by the left and right hand motor imagery. Shooting will be done using the tongue or legs. If during testing this turns out to be too difficult to interchange, it is fairly easy to set the shooting to be automatic.

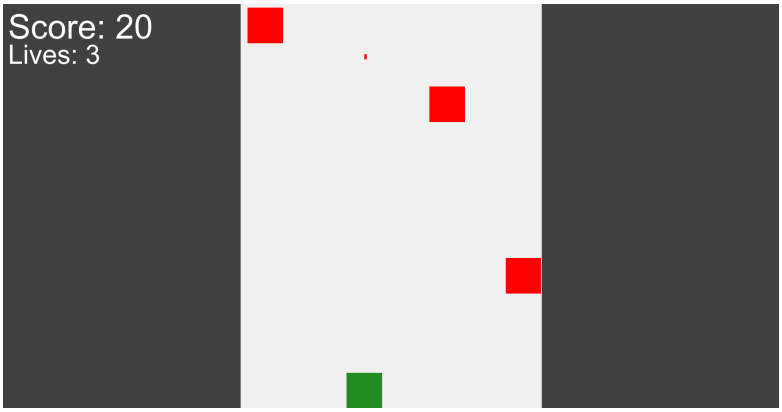


Figure 5.3: Meteor shooter

5.2.3. Conclusion

The cursor control and both games are finished, fully implemented and functional using the keyboard. The games are made taking the action latency in mind. For the floor is lava, this means the yellow, warning tiles have to take a minimal of fives seconds to turn into lava. For meteor shooter this means that the game itself is already fairly slow, and can be made even slower and easier when it turns out it is still to difficult after testing.

6 Integration

It is well known in the technological society that the most difficult and frustrating part of a project, is integrating individual parts into one whole system. Therefore, from the start, the group started communicating and making first a high level system, resulting in the overview seen in the introduction. For convenience:

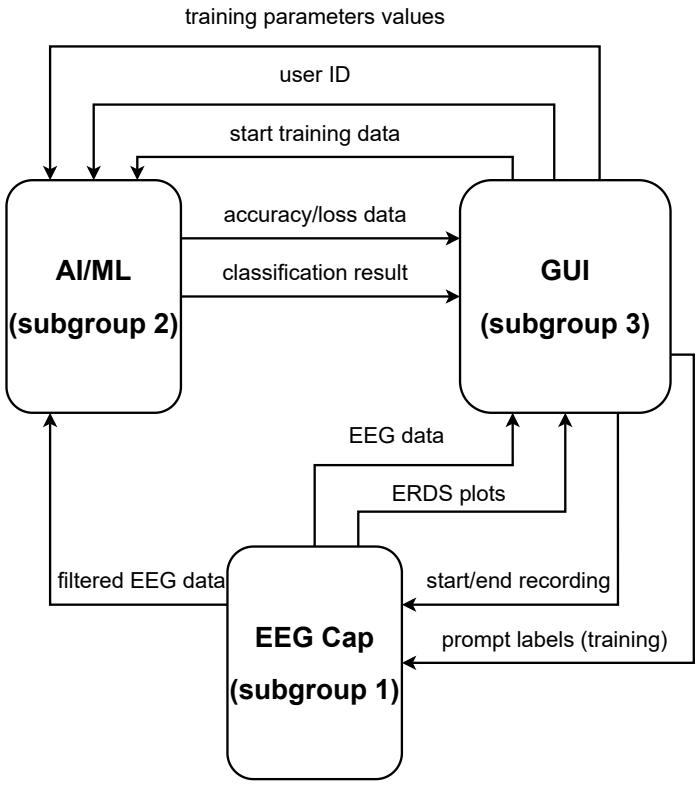


Figure 6.1: Overview of the whole system, divided into three subgroups, with the signals that are sent between them.

Knowing a high level structure, allows the subgroups to communicate with each other about the format of their input and output signals and what is expected from each other. This way the final integration will be less difficult and chaotic. Still there is always going to be some aspects which can not be anticipated. For example, the buffer of the pyLSL library discussed in chapter 4. Hence each subgroup made agreements with each other, imposing limitations and formats of their respective subgroups.

6.1. Preprocessing subgroup

The preprocessing subgroup (1) is responsible for measuring EEG data and transmitting it to the other subgroups. Hence the arrangements made, would influence the plotting of the data and the training phase, in which the EEG signals would be measured and recorded for the decoding subgroup (2). A list of some of the imposed arrangements:

- FFT, band power and ERDS will be calculated and filtered by the preprocessing subgroup.
- The EEG data sent, including for FFT and band power, will be in the format of a numpy array.

- The data for the ERDS plot will be in the format of seaborn facetGrid object (dataframe with eight columns for each channel and four rows for different frequency bands, total 32 subplots).
- Recording cycle must:
  - have a stable, consistent and as big as possible recording window.
  - contain prompt/commands and between each one a calibration cross.
  - have six seconds between cross and prompt (action latency should not exceed five seconds plus one second margin).
  - show each command six times (hence $4 * 6 = 24$ commands in total in one training cycle). This was also discussed with the client.
- Way of synchronizing time of appearance of a prompt with the respective data samples is through the subprocess library, to ensure the measuring code will run independently of the GUI code.
- GUI must contain a figure where the names of the electrodes are changed into their respective channel numbers.

6.2. Decoding subgroup

Decoding subgroup is responsible for training the recorded data and classify the MI signals. These classifications can then be used for digital actions, to control the demos and games. Hence the collaboration will have influence on the training cycle after recording the data and the classified signals to digital actions. This includes training parameters, plots and the digital actions. The accords:

- Four training parameters that can be modified using the GUI: learning rate, batch size, max iteration and margin.
- There are four classified MI signals.
- All these values will be sent as integers in python.
- For the MI signals the integers are as follow: 0 is left hand, 1 is right hand, 2 is feet and 3 is tongue.
- GUI must contain accuracy and loss plots.
- These will be supplied by the decoding subgroup in the format of numpy arrays.

6.3. Integration order

The order of integration of the subgroups is decided to be the preprocessing group with interface group, then the preprocessing group with the decoding group and lastly the decoding group with the interface group. The minimal requirement of the app is the possibility to distinguish different MI signals. The minimal way of achieving this is by plotting the EEG data. Therefore integrating the preprocessing with the interface should be done first. For the decoding subgroup to step up from online data to our own recorded data, the preprocessing subgroup would have to integrate with the decoding subgroup. This would give the decoding subgroup a chance to finetune their ML model to our own data. Parallel to that the decoding subgroup could start integrating with the interface subgroup to train the recorded data entirely from the interface. Finally the whole system can be finalized by integrating the classifications with digital actions, bringing the project to it's climax: actually make the demos and games work using the EEG signals of the user, showing that it is possible to distinguish MI signals and use it to practical and beneficial use.

A reminder that these are only the arrangements made for integration. The state of integration and the plans before the end of the BAP are described in chapter 8.

7 The Interface

As laid out in this thesis, the interface will bring all the aspects of the BCI system together, from the machine learning algorithm to the measuring and preprocessing of the data. An important aspect to always keep in mind during the project, is to plan for the integration. Meaning that making the interface is not only the shapes, buttons and colors the user sees, but also having the connections between the other modules and the backend code for integrating the modules ready.
In this chapter the interface will be explored, using the building blocks that were discussed in the previous chapters.

7.1. Main Window

Upon opening the app, a loading screen will appear until the cap is correctly connected to the laptop. If there is no cap connection, a warning will appear giving the option to reconnect or run on simulated data. If chosen to run on simulated data, it is still possible to reconnect with the cap afterwards. This fulfills requirement C1, the interface can connect and use the data of the g.tec Unicorn Hybrid Black.

7.1.1. User Page

When the app is finally open, the user page pops up. In this window the user can be selected, added, edited, etcetera (see Figure 7.1). After selecting the user, the app can be used to its full potential.

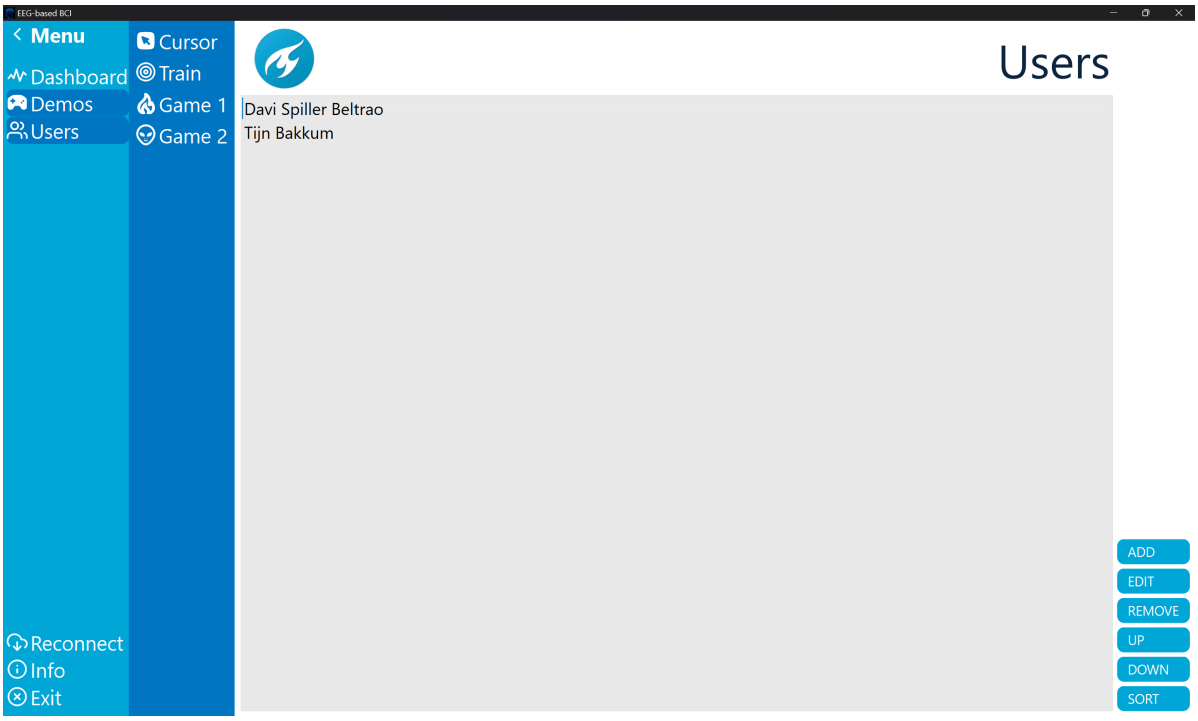


Figure 7.1: Main Window of the Graphical User Interface visualizing the User page.

In the background, the interface selects the trained ML model to the specific user from a database using their user ID. If the user doesn't have one yet, they can record and train it later. If the user chooses not to train the ML model specifically for them, a standard model will be used when using the demos. This is not recommended, since the performance of the demos won't be optimized. This fulfills requirement A4 about data handling, individualized to each user. Additionally it helps improving the sensitivity

(B1) and average overall accuracy of prediction (B2).

When clicking on the 'Dashboard' page in the left menu, the dashboard page will be shown with the name of the selected user on top.

7.1.2. Dashboard Page

On the left is the expandable menu that contains buttons to navigate through the app, see Figure 7.1.

- Dashboard is to navigate to the dashboard page.
- Demos is to open the sub menu containing the possible demos.
- Users is to navigate to the user page.
- Reconnect is to reconnect with the cap.
- Info gives information about the app and other useful information.
- Exit exits the app.

When clicking on 'Demos', a sub menu slides open with the options of the available demos. The demos will be displayed on the Users Window or the games will pop up. The options are:

- Cursor demo.
- Recording/Train user.
- Game 1 demo.
- Game 2 demo.

This accomplishes the requirement of having interactive demonstrations to visualize the decoded MI-EEG signals. In order to fulfill requirement A2, the integration with the decoding subgroup has to be finalized. More on this in section 7.2.

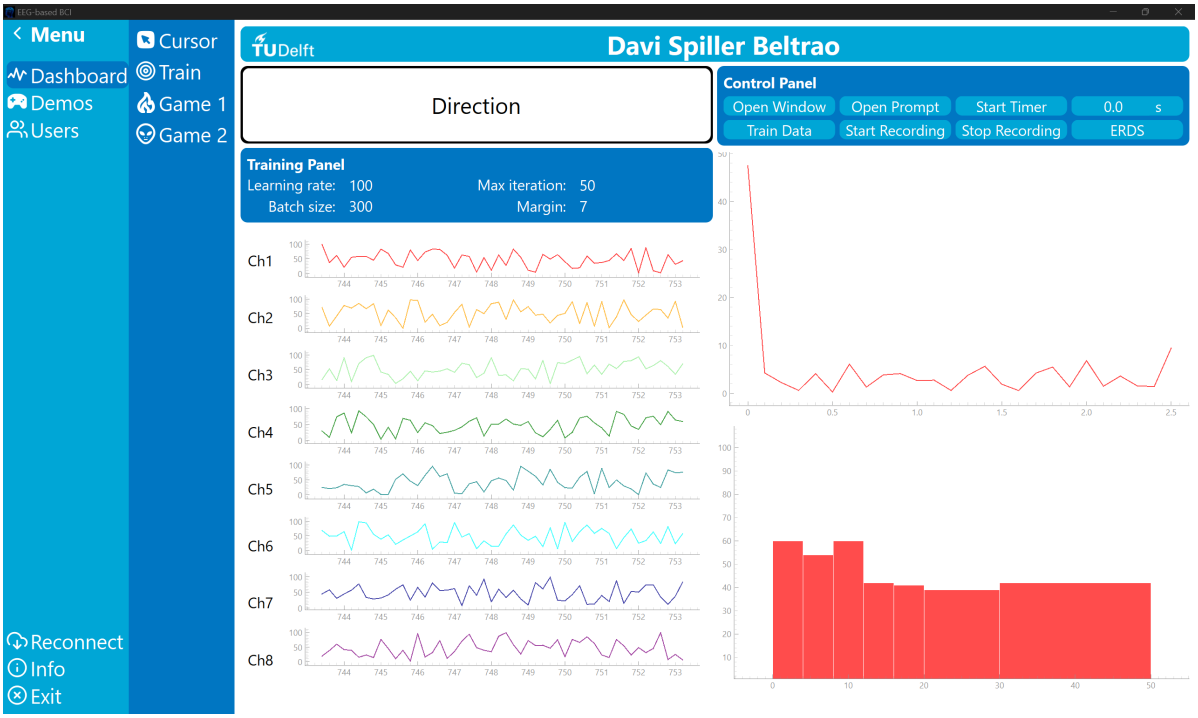


Figure 7.2: The Dashboard page on the Main Window displaying the FFT and band power plots of channel 1.

When looking at Figure 7.2, the direction box in the top of the left container displays the output of the classification (left hand, right hand, tongue, feet). This will be received from the machine learning model. Underneath it, there are training values that can be changed. Whenever these values are changed, it changes the parameters of the training model of the decoding part. The rest of the left container incorporates the real-time plots of the eight channels. The data is received from the measurement subprocess and is displayed in real-time. This subject is explored in chapter 4, fulfilling the requirement of plotting EEG data in real-time (A3), satisfying the plot latency of 1 second (B4). The plot latency is in an absolute sense negligible.

The right container accommodates from top to bottom: The control panel, buttons to control the User Window, the FFT plot and finally the band power plot. The control panel is responsible for everything that happens in the User Window. From starting and stopping of recording to measure the action latency. Specifically, the 'Train Data' button changes the FFT and band power plots into the accuracy and loss plots for the machine learning model. Finally, the 'ERDS' button calls for a window in which the ERDS plots are displayed.
The FFT and band power plots are channel dependent. The FFT and power band plots of other channels can be visualized by clicking on the keys from 1-8, according to the desired channel. These plots are also displayed in real-time.

The colors used in the Main Window are inspired by the color pallet of the TU Delft.

7.2. User Window

The User Window is the displayed window for the user. The Main Window and User Window give flexibility to work the best way possible. Preferable, the user sits still and relaxed. Having a second person coordinating everything through the Main Window allows the user to focus on the task at hand. Therefore a second monitor would be most practical. But if the user is alone, they can still do it by themselves.
Whenever the user doesn't have two screens, complicating navigation between windows, there is a possibility to double click on the control panel. This will shrink the Main Window around the control panel and the demos menu, facilitating using the User Window with only one screen.

The User Window has an arsenal of features and demos:

- Cursor page: in this page the user can move the cursor. Before implementing with the other subgroups, the user is able to move the cursor with the keys WASD. This page allows the user to test the trained ML model and see if it allows the desired distinguishability between the MI signals. This page can also be adapted to a second recording phase, where the user has neurofeedback.
- Train page: in order to test the MI signals, first the user should record their EEG data in order to train the machine learning model. This is done on this page. Using the control panel on the Main Window, the procedure can start. The procedure is as follows: if the User Window isn't open yet, open it using the button 'Open Window' on the control panel. Select the 'Train' page in the demos sub menu. A calibration cross will appear. After sitting comfortably and relaxed in order to not move to make sure the recording is as clean as possible, the recording can begin. After clicking on 'Start Recording', each six seconds will appear a new page. There are five pages, one calibration page and four MI commands/prompts: right hand, left hand, tongue and feet. Between each MI command, the calibration cross will appear. During the six seconds of the prompt, think about a specific movement with the mentioned body part(s). For even better measurement, try not to blink or move the eyes whenever the command is shown. But don't forget to breath! During the calibration cross it's not a problem, since the relevant data for training is only during the prompts. This will continue until the prompts are done.
  In the background, when starting recording, the time is synchronized with the measurement data of the cap, in order to label the data samples with their specific command/prompt. After recording is done, the data will be imported in the machine learning model for further training on the Main Window.

- Prompt page: this prompt page is accessed through the control panel on the Main Window. It's function is to test in a simple and fast way after recording and training, the time it takes to get from the measurement of the cap to displaying the action on the interface (the action latency). After clicking on 'Start Timer' five seconds will pass with the calibration cross and a prompt will appear, 'Right Hand'. The timer starts and the user is going to think about a movement with his right hand. When the ML model receives the data and decodes it, the classified action will be displayed on the 'Direction' box in the Main Window and the timer will stop, giving a hint of the performance of the system, the action latency.
- Games: other demos to demonstrate the possibility of distinguishing MI signals, is through games. Two games were made :
  - Floor is lava is a game where the user has to move the cursor away from the about-to-appear red tiles. If the red tiles appear and the cursor is on top of them, the user loses.
  - Meteor Shooter. A game where the user can move left and right and will try to shoot meteors out of the sky. When 3 meteors get to the player, the user loses.

  These can also be used for a second phase in recording and training the ML model, using neuro-feedback.

Examples of these pages are shown on the next page in Figure 7.3, Figure 7.4 and chapter 5. The calibration cross and prompt pages are made dark themed, in order to no aggravate the eyes of the user. Since it takes about five minutes to record, while having a light themed recording phase, some began having a bit of a headache and some even became nauseous. Might also be due to the lack of breathing. Which reinforces the need for the user to be trained for recording. The demos work with the WASD keys, finally achieving the requirement that the interface should work independently of the integration of the other subgroups. Due to the backend ready, the integration can be finalized to fulfill requirement A2, where the demos work with the decoded signals.

Figures of the entire GUI with all possible setups and widgets are shown in chapter 11.

7.3. Code

In order to keep consistent with the requirements, this GUI had to be made with python for an easier integration with the other subgroups. As described in chapter 4, the key aspect for this subgroup is speed. So after analyzing reviews about the different libraries for GUI and asking experienced people, the conclusion was that the library PyQt was the best for the requirements of the app. It is much faster than for example tkinter, which is also known. Besides, PyQt has a feature called Qt Designer that makes creating the components of the GUI much easier. Another advantage of PyQt is the library PyQtGraph. The benefits for these are discussed in chapter 4.

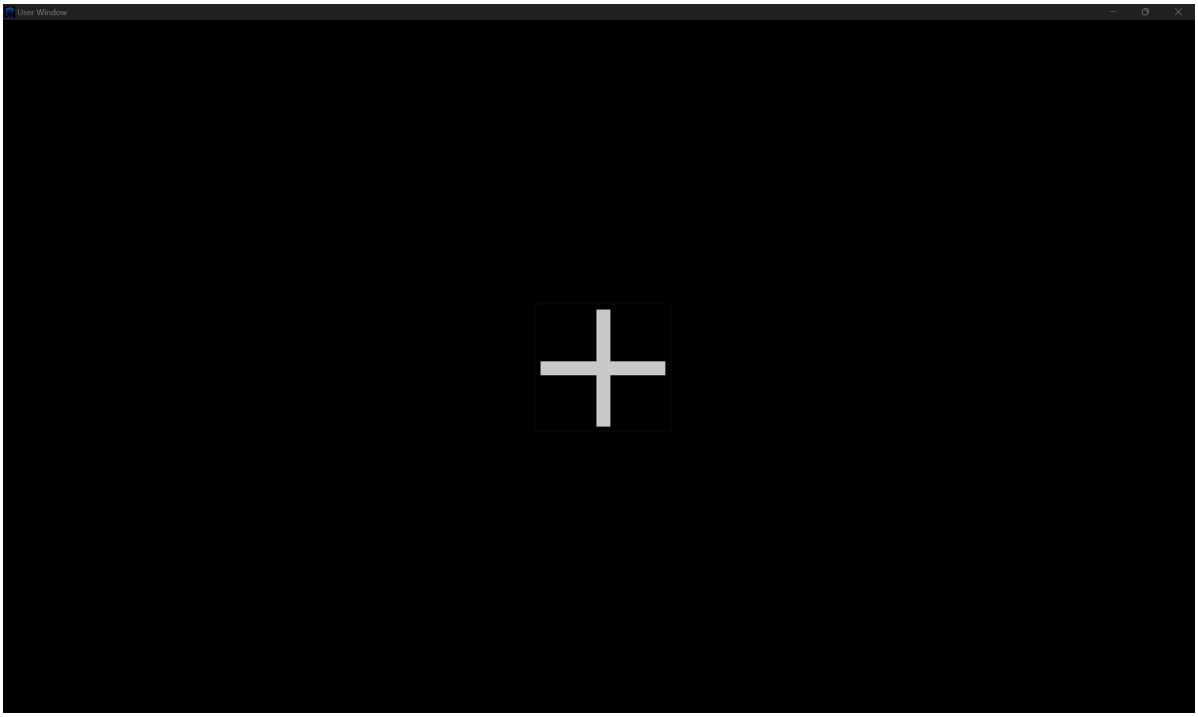


Figure 7.3: Calibration cross for train and prompt test page.

Figure 7.4: Prompt (right hand) for train and prompt test page.

8 Conclusion

8.1. Finalizing the project

To begin, a lot of the requirements of chapter 2 are fulfilled, but not yet all.
To start, the integration with the preprocessing subgroup still needs to be finished. Even though the layout is ready for the FFT, power band and ERDS plots, the time has not yet been made to make it possible to receive the plots and visualize them. In the first instance, the interface subgroup was responsible for the FFT plot. However, in order to stay consistent with the data type and handling, the choice was made to hand the responsibility over to the preprocessing subgroup. Nevertheless, the preprocessing group has made advancements, making it possible to finalize the data streaming and plotting completely.
Integrating with the decoding group hasn't even started yet, since it's the last order of integration and they are finalizing the integration with the preprocessing subgroup. The integration includes the transmission of integers and two numpy arrays for the accuracy and loss plots (see chapter 6). Since these tend to be fairly easy to implement and debug in python and with the backend code ready, the final integration can be concluded.

8.2. Improvements

Improvements within the interface are also on the lookout.
Firstly, as mentioned in section 3.2, it would be better for accomplishing the goal of this project, to add a second phase of recording data. As of now, the recording takes place in the train page of the User Window using only prompts. A possible next phase, would be a more interactive phase using neurofeedback. For example using the cursor page of the User Window in combination with prompts. That way the data can be labeled, due to the prompts, and the user can adapt to due to the neurofeedback received.
Secondly, it would be more useful to use the actual cursor of the computer instead of simulating it. Fortunately, python makes this integration easy. Additionally for the future, it would be even more handy to add a fifth MI signal, functioning as clicking.
Thirdly, to further improve the oversight of the system, a dynamic picture could be added of the cap, indicating which electrodes have a good, weak or bad connection with the scalp. Extending this idea further, topographic maps of the brain could be made to increase the possibility of distinguishing MI signals and give neurofeedback to the user.
Lastly, a button could be added saying the ERDS plots are bad. Meaning that the recorded data was bad and therefore delete the data, making it possible to record again for a better ML model.

8.3. Future plans

As for future plans, a few things could be added.
The user-based system could be enhanced by adding automatic user identification. As chapter 3 elaborated, each person has different EEG signal patterns. Using EEG biometrics to identify people has been majorly researched, since other biometrics like fingerprint and face-recognition can be 'easily' faked or copied. Especially for security reasons, a lot of investigation is being done in this area [11], [13]. They even use the same type of EEG signals (MI) used in this report [30]. There are already machine learning models made for this [3] and also using MI tasks [6].
This identification could also be a feature in the interface, to simplify things for the user. Most of the identification using EEG biometrics takes over a minute, but some research has been done using short trials, under 10 seconds [4]. For first time though, data will have to be recorded and it takes roughly five minutes. This way, a new procedure could be added whenever there is a new user using

the app. Or in the case of an MI-based ML model, during recording phase, where both models could be trained simultaneously. When adding a new user, a five-minutes-long procedure could be added so that whenever they return, they won't have to keep selecting a user.
Even further in the future, we hope this non-clinical, consumer-grade technology can be used in more advanced and practical uses. Research has already started in many applications such as predicting and preventing epilepsy[17] or rehabilitation after strokes[9] or more practical like controlling a wheelchair [21] or a character using VR headset as mentioned in chapter 5. The possibilities are endless.

8.4. Conclusion

To recap, the main goal of this project is to make an interface that is part of a BCI system that shows the possibility to distinguish different MI signals. Specifically left hand, right hand, tongue and feet. The minimal way of showing it, is by plotting the EEG data in the form of simple eight-channels real-time plot, FFT plot and band power plot. The interface plots the eight channel graphs in real-time, in order to detect possible errors and the different MI signals. Additionally, using demos and games, these MI signals can be used for practical uses. These demos work with the keys WASD.
Unfortunately, the FFT, power band and ERDS plots are not yet integrated with the EEG cap. Additionally, the integration with the decoding subgroup hasn't started. But the backend code is ready for integration and steps are made to make this possible before the final exams.

In conclusion, the main goal of this project is achieved, using the minimal way possible. This confirms the fulfillment of plotting in real-time (A3), the plot latency (B3), the implementation requirements of connection with the EEG cap (C1) and finally that the software was written in python (C2). The system functions independently, but integration with the other subgroups has to be finished in order to achieve the remaining requirements.

References

[1] Sarah N. Abdulkader, Ayman Atia, and Mostafa-Sami M. Mostafa. "Brain computer interfacing: Applications and challenges". In: *Egyptian Informatics Journal* 16.2 (2015), pp. 213–230. ISSN: 1110-8665. DOI: https://doi.org/10.1016/j.eij.2015.06.002. URL: https://www.sciencedirect.com/science/article/pii/S1110866515000237.

[2] Ahmad Chaddad et al. *Electroencephalography Signal Processing: A comprehensive review and analysis of methods and Techniques*. July 2023. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10385593/#:~:text=EEG%20is%20a%20signal%20pattern,electrodes%20applied%20onto%20the%20scalp..

[3] J. X. Chen et al. "EEG-based biometric identification with convolutional neural network". In: *Multimedia Tools and Applications* 79.15–16 (Feb. 2019), pp. 10655–10675. DOI: 10.1007/s11042-019-7258-4.

[4] Ga-Young Choi et al. "Biometrics based on single-trial EEG". In: *2019 7th International Winter Conference on Brain-Computer Interface (BCI)* (Feb. 2019). DOI: 10.1109/iww-bci.2019.8737254.

[5] W.T. Cochran et al. "What is the fast Fourier transform?" In: *Proceedings of the IEEE* 55.10 (1967), pp. 1664–1674. DOI: 10.1109/PROC.1967.5957.

[6] Essam Debie, Nour Moustafa, and Athanasios Vasilakos. "Session invariant EEG signatures using elicitation protocol fusion and Convolutional Neural Network". In: *IEEE Transactions on Dependable and Secure Computing* 19.4 (July 2022), pp. 2488–2500. DOI: 10.1109/tdsc.2021.3060775.

[7] J. A. van Deursen et al. "Increased EEG gamma band activity in Alzheimer's disease and mild cognitive impairment". In: *Journal of Neural Transmission* 115.9 (2008), pp. 1301–1311. DOI: 10.1007/s00702-008-0083-y.

[8] g.tec. *UNICORN HYBRID BLACK*. 2019. URL: https://www.gtec.at/product/unicorn-hybrid-black/.

[9] Wen Gao et al. "Application of a brain–computer interface system with visual and motor feedback in limb and brain functional rehabilitation after stroke: Case report". In: *Brain Sciences* 12.8 (Aug. 2022), p. 1083. DOI: 10.3390/brainsci12081083.

[10] Stanimira Georgieva et al. "Toward the Understanding of Topographical and Spectral Signatures of Infant Movement Artifacts in Naturalistic EEG". In: *Frontiers in Neuroscience* 14 (2020). DOI: 10.3389/fnins.2020.00352.

[11] Barjinder Kaur, Dinesh Singh, and Partha Pratim Roy. "A study of EEG for Enterprise Multimedia Security". In: *Multimedia Tools and Applications* 79.15–16 (Jan. 2020), pp. 10805–10823. DOI: 10.1007/s11042-020-08667-2.

[12] Turan A Kayagil et al. "A binary method for simple and accurate two-dimensional cursor control from EEG with minimal subject training". In: *Journal of NeuroEngineering and Rehabilitation* 6.1 (May 2009). DOI: 10.1186/1743-0003-6-14.

[13] Wanzeng Kong et al. "EEG fingerprints: Phase synchronization of EEG signals as biomarker for subject identification". In: *IEEE Access* 7 (Jan. 2019), pp. 121165–121173. DOI: 10.1109/access.2019.2931624.

[14] Ling Li. "The Differences among Eyes-Closed, Eyes-Open and Attention States: An EEG Study". In: *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. 2010, pp. 1–4. DOI: 10.1109/WICOM.2010.5600726.

[15] Fernando H. Lopes da Silva and Ernst Niedermeyer. *Electroencephalography: Basic principles, clinical applications, and related fields*. Lippincott Williams Wilkins, 2005.

[16] K.L. Lopez et al. “Stability, change, and reliable individual differences in electroencephalography measures: A lifespan perspective on progress and opportunities”. In: *NeuroImage* 275 (Apr. 2023), p. 120116. DOI: 10.1016/j.neuroimage.2023.120116.

[17] Vladimir Maksimenko et al. “Brain-computer interface for the epileptic seizures prediction and prevention”. In: *2020 8th International Winter Conference on Brain-Computer Interface (BCI)* (Feb. 2020). DOI: 10.1109/bci48061.2020.9061655.

[18] H. Marzbani, H. Marateb, and M. Mansourian. “Methodological note: Neurofeedback: A comprehensive review on system design, methodology and clinical applications”. In: *Basic and Clinical Neuroscience Journal* 7.2 (Apr. 2016). DOI: 10.15412/j.bcn.03070208.

[19] Andrew Melnik et al. “Systems, subjects, sessions: To what extent do these factors influence EEG data?” In: *Frontiers in Human Neuroscience* 11 (Mar. 2017). DOI: 10.3389/fnhum.2017.00150.

[20] Valerie Morash et al. “Classifying EEG signals preceding right hand, left hand, tongue, and right foot movements and motor imageries”. In: *Clinical Neurophysiology* 119.11 (Nov. 2008), pp. 2570–2578. DOI: 10.1016/j.clinph.2008.08.013.

[21] Mohammad Y. Naser and Sylvia Bhattacharya. “Towards practical BCI-driven wheelchairs: A systematic review study”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 31 (Jan. 2023), pp. 1030–1044. DOI: 10.1109/tnsre.2023.3236251.

[22] Kang Pan et al. “A noninvasive BCI system for 2D cursor control using a spectral-temporal long short-term memory network”. In: *Frontiers in Computational Neuroscience* 16 (Mar. 2022). DOI: 10.3389/fncom.2022.799019.

[23] G. Pfurtscheller and F.H. Lopes da Silva. “Event-related EEG/MEG synchronization and desynchronization: basic principles”. In: *Clinical Neurophysiology* 110.11 (1999), pp. 1842–1857. ISSN: 1388-2457. DOI: https://doi.org/10.1016/S1388-2457(99)00141-8. URL: https://www.sciencedirect.com/science/article/pii/S1388245799001418.

[24] Parmar Prashant, Anand Joshi, and Vaibhav Gandhi. “Brain computer interface: A review”. In: *2015 5th Nirma University International Conference on Engineering (NUiCONE)*. 2015, pp. 1–6. DOI: 10.1109/NUICONE.2015.7449615.

[25] Shadi Sartipi and Mujdat Cetin. “Subject-independent deep architecture for EEG-based motor imagery classification”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 32 (Jan. 2024), pp. 718–727. DOI: 10.1109/tnsre.2024.3360194.

[26] Sccn. *SCCN/labstreaminglayer: Labstreaminglayer super repository comprising submodules for LSL and associated apps.* URL: https://github.com/sccn/labstreaminglayer.

[27] Kaoru Sumi et al. “A Cooperative Game Using the P300 EEG-Based Brain-Computer Interface”. In: *Assistive and Rehabilitation Engineering*. Ed. by Yves Rybarczyk. Rijeka: IntechOpen, 2019. Chap. 10. DOI: 10.5772/intechopen.84621. URL: https://doi.org/10.5772/intechopen.84621.

[28] Michele Terzaghi et al. “Sleep paralysis in narcolepsy: More than just a motor dissociative phenomenon?” In: *Neurological sciences : official journal of the Italian Neurological Society and of the Italian Society of Clinical Neurophysiology* 33 (June 2011), pp. 169–72. DOI: 10.1007/s10072-011-0644-y.

[29] Gabriel Alves Mendes Vasiljevic and Leonardo Cunha de Miranda. “Brain–Computer Interface Games Based on Consumer-Grade EEG Devices: A Systematic Literature Review”. In: *International Journal of Human–Computer Interaction* 36.2 (2020), pp. 105–142. DOI: 10.1080/10447318.2019.1612213. eprint: https://doi.org/10.1080/10447318.2019.1612213. URL: https://doi.org/10.1080/10447318.2019.1612213.

[30] Yichuan Wang et al. “Never lost keys: A novel key generation scheme based on motor imagery EEG in end-edge-cloud system”. In: *China Communications* 19.7 (July 2022), pp. 172–184. DOI: 10.23919/jcc.2022.07.014.

[31] Jonathan R. Wolpaw et al. “An EEG-based brain-computer interface for cursor control”. In: *Electroencephalography and Clinical Neurophysiology* 78.3 (1991), pp. 252–259. ISSN: 0013-4694. DOI: https://doi.org/10.1016/0013-4694(91)90040-B. URL: https://www.sciencedirect.com/science/article/pii/001346949190040B.

[32] Han Yuan and Bin He. “Brain–computer interfaces using sensorimotor rhythms: Current state and future perspectives”. In: *IEEE Transactions on Biomedical Engineering* 61.5 (May 2014), pp. 1425–1435. DOI: 10.1109/tbme.2014.2312397.

9 Source Code

In this appendix, the source code of the interface is stretched out. The file worked on was only the main.py file. The rest was delivered by the Qt Designer and nothing was changed.

9.1. MAIN.PY

```
import os
import sys
import json
import csv
import subprocess
from ui_interface import *
from ui_splashscreen import *
from ui_ERDSWindow import Ui_ERDSWindow
from ui_userWindow import Ui_UserWindow
from Custom_Widgets import *
from PySide6.QtWidgets import QInputDialog, QMessageBox, QSplashScreen, QFrame
from PySide6.QtCore import Qt, QTimer, Slot, Signal, QEvent, QBasicTimer
from PySide6.QtGui import QPainter, QColor, QFont
import random
import numpy as np
import pandas as pd
import time
import pyqtgraph as pg
from pylsl import StreamInlet, resolve_stream

#====================================================================
# Initialization Splashscreen
#====================================================================
class SplashScreen(QSplashScreen):
    def __init__(self):
        super(SplashScreen, self).__init__()
        self.ui = Ui_SplashScreen()
        self.ui.setupUi(self) # Load UI from ui_splashscreen.py
        self.setWindowFlags(Qt.WindowType.FramelessWindowHint | Qt.WindowType.
    WindowStaysOnTopHint)
        self.setAttribute(Qt.WidgetAttribute.WA_TranslucentBackground) # Make window
    frameless and without background

        # Initialize progress bar for duration of splashscreen
        self.ui.progressBar.setMinimum(0)
        self.ui.progressBar.setMaximum(100)

        # Set up a timer to update the progress bar
        self.progress_timer = QTimer()
        self.progress_timer.timeout.connect(self.update_progress)
        self.progress_timer.start(100)

        # Counter for tracking progress
        self.progress_value = 0

    def update_progress(self):
        # Increment progress value
        self.progress_value += 1
        self.ui.progressBar.setValue(self.progress_value)

        # Check if progress is complete
        if self.progress_value >= 100:
            self.progress_timer.stop()

    # Show screen until closed
    def showEvent(self, event):
```

```
55          super().showEvent(event)
56          self.centerSplash()
57
58      # Center the splashscreen to the monitor screen
59      def centerSplash(self):
60          screen = self.screen()
61          screen_geometry = screen.geometry()
62          splash_geometry = self.geometry()
63
64          x = (screen_geometry.width() - splash_geometry.width()) // 2
65          y = (screen_geometry.height() - splash_geometry.height()) // 2
66
67          self.move(x, y)
68
69  #=========================================================================
70  # Main Window from which everything can be called
71  #=========================================================================
72  class MainWindow(QMainWindow):
73      #Signals to send to the Users Window to change the page
74      userWindow_to_cursorPage = Signal()
75      userWindow_to_trainingPage = Signal()
76      userWindow_to_promptPage = Signal()
77      userWindow_startRecording = Signal()
78      userWindow_stopRecording = Signal()
79      userWindow_startPromptTimer = Signal()
80
81      def __init__(self):
82          super(MainWindow, self).__init__()
83
84          self.startFFT = False
85          self.done_recording = False
86
87          # For EEG cap connection and data
88          self.simulate_data = False
89          self.streams = resolve_stream()
90          try:
91              self.inlet = StreamInlet(self.streams[0])
92          except:
93              self.show_eeg_error("The EEG cap is not connected. Please connect the cap.")
94
95          self.ui = Ui_MainWindow()
96          self.ui.setupUi(self) # Import UI from ui_interface.py
97
98          # Set the control panel
99          self.controlPanel = self.findChild(QWidget, "buttonsBox")
100         self.controlPanel.installEventFilter(self)
101
102         self.setWindowTitle("EEG-based BCI")
103
104         # Apply style from the file style.json
105         loadJsonStyle(self, self.ui, jsonFiles = {
106                         "logs/style.json"
107                             })
108
109         # Predefined colors
110         colors = [
111             QColor(255, 0, 0),    # Red
112             QColor(255, 165, 0),  # Orange
113             QColor(144, 238, 144),# Light Green
114             QColor(0, 128, 0),    # Green
115             QColor(0, 128, 128),  # Blue-Green
116             QColor(0, 255, 255),  # Cyan
117             QColor(0, 0, 139),    # Dark Blue
118             QColor(128, 0, 128),  # Purple
119             QColor(255, 0, 255),  # Magenta
120             QColor(255, 255, 0),  # Yellow
121             QColor(0, 255, 0),    # Green
122             QColor(0, 0, 255)     # Blue
123         ]
124
125         # Convert to pastel colors
```

```
126         self.pastel_colors = [self.make_pastel(color) for color in colors]
127 
128         #Check if the buttons are clicked and evoke their function
129         #User page:
130         self.ui.addBtn.clicked.connect(self.addUser)
131         self.ui.editBtn.clicked.connect(self.editUser)
132         self.ui.removeBtn.clicked.connect(self.removeUser)
133         self.ui.upBtn.clicked.connect(self.upUser)
134         self.ui.downBtn.clicked.connect(self.downUser)
135         self.ui.sortBtn.clicked.connect(self.sortUser)
136         self.ui.usersList.itemClicked.connect(self.ChooseUser)
137         #Menu
138         self.ui.reconnectBtn.clicked.connect(self.reconnect_cap)
139         self.ui.overviewBtn.clicked.connect(self.changeOverviewBtn)
140         self.ui.usersBtn.clicked.connect(self.changeUsersBtn)
141         self.ui.demosBtn.clicked.connect(self.changeDemosBtn)
142         self.ui.exitBtn.clicked.connect(self.exitApp)
143         #Demos submenu
144         self.ui.cursorBtn.clicked.connect(self.setCursorPage)
145         self.ui.trainBtn.clicked.connect(self.setTrainPage)
146         self.ui.game1Btn.clicked.connect(self.openLavaGame)
147         self.ui.game2Btn.clicked.connect(self.openAsteroid)
148         #Button panel
149         self.ui.startRecordingBtn.clicked.connect(self.startRecording)
150         self.ui.stopRecordingBtn.clicked.connect(self.stopRecording)
151         self.ui.openUserWindowBtn.clicked.connect(self.openUserWindow)
152         self.ui.ERDSBtn.clicked.connect(self.openERDSWindow)
153         self.ui.openPromptBtn.clicked.connect(self.setPromptPage)
154         self.ui.startTimerBtn.clicked.connect(self.startPromptTimer)
155 
156         # add users to user list from file
157         with open('users.csv', newline='') as user_file:
158             user_reader = csv.DictReader(user_file)
159             for row in user_reader:
160                 currentIndex = self.ui.usersList.currentRow()
161                 self.ui.usersList.insertItem(currentIndex, row["Name"])
162 
163 
164         #Data live plotting
165         self.i = 0
166         self.j = 0
167         self.max_graph_width = 50
168         self.plot_update_size = 2
169         self.columns = 7
170         self.av_height = int(self.max_graph_width/self.columns)
171         self.channel = 1
172 
173         self.xdata = np.zeros(self.max_graph_width)
174         self.ydata = [np.zeros(self.max_graph_width) for _ in range(8)]
175         self.yBarGraph = np.zeros(self.columns)
176         symbol_sign = None
177 
178         # ML plots
179         self.accuracy_data = np.zeros(1)
180         self.accuracy_data_iter = np.zeros(1)
181         self.loss_data = np.array([100])
182         self.loss_data_iter = np.zeros(1)
183 
184         # Create subplots and lines
185         self.subplots = []
186         self.lines = []
187 
188         self.j = 0
189 
190         self.ui.channelsPlot.setBackground(QColor(255, 255, 255))
191 
192         for i in range(8):
193             p = self.ui.channelsPlot.addPlot(row=i, col=0)
194             p.setMouseEnabled(x=False, y=False)
195             p.setMenuEnabled(False)
196             export = self.ui.channelsPlot.sceneObj.contextMenu
```

```
197             del export[:]
198             p.hideButtons()
199             self.subplots.append(p)
200             self.lines.append(p.plot(pen=pg.mkPen(self.pastel_colors[i], width = 2)))
201             #p.hideAxis('bottom')
202             #p.hideAxis('left')
203 
204         # Bar graph band power
205         self.xBarGraph = np.array([2,6,10,14,18,25,40]) #Center points of the columns with
    according width /<--
206         self.power_band_1= pg.BarGraphItem(x=self.xBarGraph[[0,1,2,3,4]], height = self.
    yBarGraph[[0,1,2,3,4]], width = 4, brush = QColor(0, 166, 214), pen=QColor(255, 255, 255)
    )
207         self.power_band_2 = pg.BarGraphItem(x=self.xBarGraph[[5]], height = self.yBarGraph
    [[5]], width = 10, brush = QColor(0, 166, 214), pen=QColor(255, 255, 255))
208         self.power_band_3 = pg.BarGraphItem(x=self.xBarGraph[[6]], height = self.yBarGraph
    [[6]], width = 20, brush = QColor(0, 166, 214), pen=QColor(255, 255, 255))
209         self.ui.powerBandPlot.addItem(self.power_band_1)
210         self.ui.powerBandPlot.addItem(self.power_band_2)
211         self.ui.powerBandPlot.addItem(self.power_band_3)
212         self.ui.powerBandPlot.setYRange(10, 100)
213         self.ui.powerBandPlot.setXRange(0, 50)
214         self.ui.powerBandPlot.setMouseEnabled(x=False, y=False)
215         self.ui.powerBandPlot.setMenuEnabled(False)
216         self.ui.powerBandPlot.hideButtons()
217 
218         self.start_time = time.time()
219 
220         # Add a timer to simulate new temperature measurements
221         self.timer = QTimer()
222         self.timer.timeout.connect(self.update_plot)
223         self.timer.start(0.04)
224 
225         # Stopwatch variables
226         self.count = 0
227         self.flag = False
228         self.ui.stopwatch.setText(str(self.count))
229         self.stopwatch = QTimer()
230         self.stopwatch.timeout.connect(self.showTime)
231         self.stopwatch.start(100)
232 
233         #Threshold of minimum width of window
234         self.min_width = 1000
235         self.original_geometry = self.geometry()
236 
237     # Check if double clicked on control panel, if true minimze window around the control
    panel
238     def eventFilter(self, obj, event):
239         if obj == self.controlPanel and event.type() == QEvent.MouseButtonDblClick:
240             self.minimizeWindow()
241             return True
242         return super().eventFilter(obj, event)
243 
244     def minimizeWindow(self):
245         if self.width() > self.min_width: # Check state of window
246             # Hide other components
247             self.ui.leftMenuContainer.hide()
248             self.ui.frame_2.hide()
249             self.ui.frame_3.hide()
250             self.ui.leftSubMenu.expandMenu()
251             self.ui.UserIDBox.hide()
252             self.ui.infoWidgetContainer.hide()
253             self.ui.leftBodyFrameOverview.hide()
254             self.ui.FFTFrame.hide()
255             self.ui.powerBandFrame.hide()
256 
257             # Resize the window to the size of the control panel widget
258             self.setFixedSize(900,200)
259             self.setWindowFlags(Qt.Widget | Qt.WindowStaysOnTopHint)
260 
261             # Set background to transparent
```

```
262             self.setAttribute(Qt.WA_TranslucentBackground, True)
263             self.show()
264 
265         else:
266             self.setFixedSize(1595, 831)
267             self.setMinimumSize(0, 0)
268             self.setMaximumSize(16777215, 16777215)
269 
270             # Restore window frame and background
271             self.setWindowFlags(Qt.Widget)
272             self.setAttribute(Qt.WA_TranslucentBackground, False)
273             self.setGeometry(self.original_geometry)
274             #Show other components
275             self.ui.leftMenuContainer.show()
276             self.ui.frame_2.show()
277             self.ui.frame_3.show()
278             self.ui.leftSubMenu.show()
279             self.ui.leftSubMenu.collapseMenu()
280             self.ui.UserIDBox.show()
281             self.ui.infoWidgetContainer.show()
282             self.ui.leftBodyFrameOverview.show()
283             self.ui.FFTFrame.show()
284             self.ui.powerBandFrame.show()
285 
286             self.showMaximized()
287 
288     # Make plot colors pastel
289     def make_pastel(self, color, factor=0.3):
290         white = QColor(255, 255, 255)
291         color = QColor(color)
292         return QColor(
293             int(color.red() + (white.red() - color.red()) * factor),
294             int(color.green() + (white.green() - color.green()) * factor),
295             int(color.blue() + (white.blue() - color.blue()) * factor)
296         )
297 
298     def showTime(self):
299         # checking if flag is true
300         if self.flag:
301             self.count += 1
302         else:
303             self.count = 0
304 
305         # Getting text from count
306         if self.count < 47:
307             text = "0.0"
308         else:
309             text = str(float("{:.1f}".format(self.count / 10 - 4.7)))
310 
311         # Show text
312         self.ui.stopwatch.setText(text)
313 
314     # Emit the signals to the User Window to change page
315     def setCursorPage(self):
316             self.userWindow_to_cursorPage.emit()
317 
318     def setTrainPage(self):
319             self.userWindow_to_trainingPage.emit()
320 
321     def setPromptPage(self):
322             self.userWindow_to_promptPage.emit()
323             self.ui.stopwatch.setText("0.0") # Reset timer
324             self.flag = False
325 
326     def startRecording(self):
327             self.userWindow_startRecording.emit()
328 
329     def stopRecording(self):
330             self.userWindow_stopRecording.emit()
331             self.flag = False
332 
```

```
333     def startPromptTimer(self):
334             self.flag = True
335             self.userWindow_startPromptTimer.emit()
336 
337     # Try to reconnect with cap again
338     def reconnect_cap(self):
339         try:
340             self.inlet = StreamInlet(self.streams[0], max_buflen=0)
341             dlg = QMessageBox()
342             dlg.setWindowTitle("EEG cap connected")
343             dlg.setText("The EEG cap is succesfully connected. The data shown will now be the
     real data.")
344             button = dlg.exec()
345             self.simulate_data = False
346             self.xdata = np.zeros(self.max_graph_width)
347             self.ydata = [np.zeros(self.max_graph_width) for _ in range(8)]
348             self.i = 0
349         except:
350             self.show_eeg_error("The EEG cap could not connect. Please try again.")
351 
352     # Show error if no cap connected
353     def show_eeg_error(self, error_text):
354         dlg = QMessageBox()
355         dlg.setWindowTitle("ERROR")
356         dlg.setStandardButtons(QMessageBox.StandardButton.Retry | QMessageBox.StandardButton.
    Ignore)
357         dlg.setText(error_text)
358 
359         screen = self.screen()
360         screen_geometry = screen.geometry()
361         splash_geometry = self.geometry()
362 
363         x = (screen_geometry.width() - splash_geometry.width()) // 2
364         y = 0
365 
366         dlg.move(x, y)
367         button = dlg.exec()
368 
369         if button == QMessageBox.StandardButton.Retry:
370             print("retrying....")
371             try:
372                 self.inlet = StreamInlet(self.streams[0], max_buflen=0)
373                 self.simulate_data = False
374             except:
375                 self.show_eeg_error(error_text)
376         elif button == QMessageBox.StandardButton.Ignore:
377             dlg = QMessageBox()
378             dlg.setWindowTitle("Ignored")
379             dlg.setText("The program will now run without the EEG cap data and will use
    simulated data. "
380                         "To use the EEG cap restart the program.")
381             dlg.move(x, y)
382             button = dlg.exec()
383             self.simulate_data = True
384 
385     # Update graphs
386     def update_plot(self):
387         pen = pg.mkPen(self.pastel_colors[self.channel - 1], width = 2)
388         # Gathering the data from the EEG cap
389         if not self.simulate_data:
390             sample, timestamp = self.inlet.pull_sample()
391             sample_timestamp = (self.i) / 250
392         else:
393             sample, timestamp = self.generate_random_sample()  # For testing purposes when
    not connected to cap
394             sample_timestamp = self.i / 250
395 
396         if self.i <= self.max_graph_width:
397             pass
398         else:
399             if not self.startFFT:
```

```
400                 self.startFFT = True
401                 symbol_sign = None
402                 self.FFT_plot = self.ui.FFTPlot.plot(
403                     self.xdata,
404                     self.ydata[self.channel - 1],
405                     name="Power Sensor",
406                     pen=pen,
407                     symbol=symbol_sign,
408                     symbolSize=5,
409                     symbolBrush="b",
410                 )
411                 #self.ui.FFTPlot.setXRange(5, 35)
412                 #self.ui.FFTPlot.setYRange(0, 50)
413                 self.ui.FFTPlot.setMouseEnabled(x=False, y=False)
414                 self.ui.FFTPlot.setMenuEnabled(False)
415                 self.ui.FFTPlot.hideButtons()
416                 self.FFT_plot.setFftMode(True)
417 
418         # Only update the plot everytime it has collected plot update data sized data
419         if self.i % 50 == 0:
420             self.xdata = np.roll(self.xdata, -1)
421             self.xdata[-1] = sample_timestamp
422 
423             for k in range(8):
424                 self.ydata[k] = np.roll(self.ydata[k], -1)
425                 self.ydata[k][-1] = sample[k]
426                 self.lines[k].setData(self.xdata, self.ydata[k])
427 
428             self.power_band_1.setOpts(height=self.yBarGraph[[0, 1, 2, 3, 4]], brush=pg.
    mkBrush(self.pastel_colors[self.channel - 1]))
429             self.power_band_2.setOpts(height=self.yBarGraph[[5]], brush=pg.mkBrush(self.
    pastel_colors[self.channel - 1]))
430             self.power_band_3.setOpts(height=self.yBarGraph[[6]], brush=pg.mkBrush(self.
    pastel_colors[self.channel - 1]))
431 
432             if self.startFFT:
433                 self.FFT_plot.setData(self.xdata, self.ydata[self.channel - 1])
434                 self.FFT_plot.setPen(pg.mkPen(self.pastel_colors[self.channel - 1], width =
    2))
435 
436             # Change the power band plots from channel
437             self.yBarGraph = np.array(
438                 [sum(self.ydata[self.channel - 1][i:i + self.av_height]) // self.av_height
    for i in
439                  range(0, len(self.ydata[self.channel - 1]), self.av_height)])
440 
441         self.i += 1
442 
443         if self.i == 1000:
444             print(time.time() - self.start_time)
445 
446 
447     # For testing purposes
448     def generate_random_sample(self):
449         # Simulate random data generation
450         return random.sample(range(0, 100), 15) + [time.time()], 0
451 
452     # Set and open User Window functions
453     def setUserWindow(self, userWindow):
454         self.userWindow = userWindow
455 
456     def openUserWindow(self):
457         global recProcess
458         if self.userWindow.isVisible():
459             pass
460         else:
461             recProcess = subprocess.Popen(["python3", "-u", "MeasurementSubgroup/Streaming/
    LSL_csv.py"], stdin=subprocess.PIPE, stdout=subprocess.PIPE,)
462             self.userWindow.show()
463 
464     # Set and open ERDS Window functions
```

```
465     def setERDSWindow(self, ERDSWindow):
466         self.ERDSWindow = ERDSWindow
467 
468     def openERDSWindow(self):
469         self.ERDSWindow.show()
470 
471     # Set and open Lava Game Window functions
472     def setLavaGameWindow(self, LavaGame):
473         self.LavaGame = LavaGame
474 
475     def openLavaGame(self):
476         self.LavaGame.showMaximized()
477         self.userWindow.hide()
478 
479     # Set and open Asteroid Widnow functions
480     def setAsteroidWindow(self, Asteroid):
481         self.Asteroid = Asteroid
482 
483     def openAsteroid(self):
484         self.Asteroid.showMaximized()
485         self.userWindow.hide()
486 
487     # Exit the app
488     def exitApp(self):
489         QApplication.quit()
490 
491     # Will start training the ML model on the new data
492     def start_training(self):
493         self.accuracy_data = np.append(self.accuracy_data, random.sample(range(int(self.
    accuracy_data[-1]), 100), 1))
494         self.accuracy_data_iter = np.append(self.accuracy_data_iter, self.accuracy_data_iter
    [-1] + 1)
495         self.loss_data = np.append(self.loss_data, random.sample(range(0, self.loss_data[-1]
    + 1), 1))
496         self.loss_data_iter = np.append(self.loss_data_iter, self.loss_data_iter[-1] + 1)
497         self.update_ML_plots()
498 
499     # updating the Machine Learning plots while training
500     def update_ML_plots(self):
501         # when it's the first time after recording we want to clear the previous plots
502         if self.done_recording == False:
503             pen = pg.mkPen(self.pastel_colors[self.channel - 1], width=2)
504             # if the FFT and frequency band plots were used, clear them
505             if self.startFFT:
506                 self.FFT_plot.clear()
507                 self.ui.powerBandPlot.clear()
508             symbol_sign = None
509             self.loss_plot = self.ui.powerBandPlot.plot(
510                 self.loss_data_iter,
511                 self.loss_data,
512                 name="Power Sensor",
513                 pen=pen,
514                 symbol=symbol_sign,
515                 symbolSize=5,
516                 symbolBrush="b",
517             )
518             self.ui.powerBandPlot.setYRange(0, 100)
519             self.ui.powerBandPlot.setXRange(0, int(self.ui.maxIterationLine.text()))
520             self.accuracy_plot = self.ui.FFTPlot.plot(
521                 self.accuracy_data_iter,
522                 self.accuracy_data,
523                 name="Power Sensor",
524                 pen=pen,
525                 symbol=symbol_sign,
526                 symbolSize=5,
527                 symbolBrush="b",
528             )
529             self.ui.FFTPlot.setYRange(0, 100)
530             self.ui.FFTPlot.setXRange(0, int(self.ui.maxIterationLine.text()))
531             self.done_recording = True
532 
```

```
533         # update the plots with the new data
534         self.accuracy_plot.setData(self.accuracy_data_iter, self.accuracy_data)
535         self.loss_plot.setData(self.loss_data_iter, self.loss_data)
536 
537 
538     #Functions that handles the user based interface
539     def ChooseUser(self, item):
540         if type(item) is str:
541             self.ui.userID_test.setText(item)
542             self.current_id = None
543             with open('users.csv', newline='') as file:
544                 reader = csv.DictReader(file)
545                 for row in reader:
546                     if row['Name'] == item:
547                         self.current_id = row['ID']
548             if not self.current_id:
549                 print("ERROR: USER " + item + " HAS NO CORRESPONDING ID.")
550 
551         else:
552             self.ui.userID_test.setText(item.text())
553             self.current_id = None
554             with open('users.csv', newline='') as file:
555                 reader = csv.DictReader(file)
556                 for row in reader:
557                     if row['Name'] == item.text():
558                         self.current_id = row['ID']
559             if not self.current_id:
560                 print("ERROR: USER " + item.text() + " HAS NO CORRESPONDING ID.")
561         print(self.current_id)
562 
563     # Add user name
564     def addUser(self):
565         currentIndex = self.ui.usersList.currentRow()
566         error = ""
567         while True:
568             if error:
569                 text, ok = QInputDialog.getText(None, "New User", error)
570             else:
571                 text, ok = QInputDialog.getText(None, "New User", "User Name")
572             error = ""
573             if ok and text is not None:
574                 if text == "":
575                     error = "Please fill in a valid username"
576                 with open('users.csv', newline='') as user_file:
577                     reader = csv.DictReader(user_file)
578                     highest_id = 0
579                     for row in reader:
580                         if int(row["ID"]) > highest_id:
581                             highest_id = int(row["ID"])
582                         if row["Name"] == text:
583                             error = "This user already exists"
584                 if not error:
585                     with open('users.csv', 'a+', newline='') as user_file:
586                         writer = csv.writer(user_file)
587                         writer.writerow([text, highest_id + 1])
588                     self.ui.usersList.insertItem(currentIndex, text)
589                     self.ChooseUser(text)
590                     return
591             else:
592                 return
593 
594     # Edit user name
595     def editUser(self):
596         currentIndex = self.ui.usersList.currentRow()
597         item = self.ui.usersList.item(currentIndex)
598         if item is not None:
599             error = ""
600             while True:
601                 if error:
602                     text, ok = QInputDialog.getText(None, "Change Username", error)
603                 else:
```

```
604                     text, ok = QInputDialog.getText(None, "Change Username", "User Name")
605                 error = ""
606                 if ok and text is not None:
607                     if text == "":
608                         error = "Please fill in a valid username"
609 
610                     rows = []
611                     with open('users.csv', newline='') as file:
612                         reader = csv.DictReader(file)
613                         for row in reader:
614                             if row["Name"] == text and text != item.text():
615                                 error = "This user already exists"
616                             elif row['Name'] == item.text():  # Update the name for the
    corresponding user
617                                 row['Name'] = text
618                             rows.append(row)
619                     if not error:
620                         with open('users.csv', 'w', newline='') as file:
621                             writer = csv.DictWriter(file, fieldnames=['Name', 'ID'])
622                             writer.writeheader()
623                             writer.writerows(rows)
624                         item.setText(text)
625                         self.ChooseUser(text)
626                         return
627                 else:
628                     return
629 
630     # Remove user
631     def removeUser(self):
632         currentIndex = self.ui.usersList.currentRow()
633         item = self.ui.usersList.item(currentIndex)
634         if item is None:
635             return
636 
637         question = QMessageBox.question(None,"Remove User",
638              "Do you want to remove user: " + item.text(),
639              QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No)
640 
641         if question == QMessageBox.StandardButton.Yes:
642             rows = []
643             with open('users.csv', newline='') as file:
644                 reader = csv.DictReader(file)
645                 for row in reader:
646                     if row['Name'] != item.text():  # Update the name for the corresponding
    user
647                         rows.append(row)
648             with open('users.csv', 'w', newline='') as file:
649                 writer = csv.DictWriter(file, fieldnames=['Name', 'ID'])
650                 writer.writeheader()
651                 writer.writerows(rows)
652             item = self.ui.usersList.takeItem(currentIndex)
653             del item
654             del item
655 
656     # Move user up
657     def upUser(self):
658         index = self.ui.usersList.currentRow()
659         if index >= 1:
660             item = self.ui.usersList.takeItem(index)
661             self.ui.usersList.insertItem(index-1,item)
662             self.ui.usersList.setCurrentItem(item)
663 
664     # Move user down
665     def downUser(self):
666         index = self.ui.usersList.currentRow()
667         if index < self.ui.usersList.count()-1:
668             item = self.ui.usersList.takeItem(index)
669             self.ui.usersList.insertItem(index + 1, item)
670             self.ui.usersList.setCurrentItem(item)
671 
672     # Sort user names
```

```
673     def sortUser(self):
674         self.ui.usersList.sortItems()
675 
676     # Change color of menu buttons according to the page
677     def changeOverviewBtn(self):
678         if self.ui.mainPages.currentIndex() == 0:
679             self.ui.overviewBtn.setStyleSheet("background-color: rgb(0, 118, 194);")
680             self.ui.usersBtn.setStyleSheet("background-color: rgb(0, 166, 214);")
681             self.ui.demosBtn.setStyleSheet("background-color: rgb(0, 166, 214);")
682 
683     def changeUsersBtn(self):
684         if self.ui.mainPages.currentIndex() == 1:
685             self.ui.usersBtn.setStyleSheet("background-color: rgb(0, 118, 194);")
686             self.ui.overviewBtn.setStyleSheet("background-color: rgb(0, 166, 214);")
687             self.ui.demosBtn.setStyleSheet("background-color: rgb(0, 166, 214);")
688 
689     def changeDemosBtn(self):
690         self.ui.demosBtn.setStyleSheet("background-color: rgb(0, 118, 194);")
691 
692     # Functions when clicking on keys
693     def keyPressEvent(self, event):
694         # Functions to change FFT and band power when clicking on 1-8
695         if event.key() == Qt.Key.Key_1:
696             self.yBarGraph = np.array([sum(self.ydata[0][i:i+self.av_height])//self.av_height
     for i in range(0,len(self.ydata[0]),self.av_height)])
697             self.channel = 1
698         elif event.key() == Qt.Key.Key_2:
699             self.yBarGraph = np.array([sum(self.ydata[1][i:i+self.av_height])//self.av_height
     for i in range(0,len(self.ydata[1]),self.av_height)])
700             self.channel = 2
701         elif event.key() == Qt.Key.Key_3:
702             self.yBarGraph = np.array([sum(self.ydata[2][i:i+self.av_height])//self.av_height
     for i in range(0,len(self.ydata[2]),self.av_height)])
703             self.channel = 3
704         elif event.key() == Qt.Key.Key_4:
705             self.yBarGraph = np.array([sum(self.ydata[3][i:i+self.av_height])//self.av_height
     for i in range(0,len(self.ydata[3]),self.av_height)])
706             self.channel = 4
707         elif event.key() == Qt.Key.Key_5:
708             self.yBarGraph = np.array([sum(self.ydata[4][i:i+self.av_height])//self.av_height
     for i in range(0,len(self.ydata[4]),self.av_height)])
709             self.channel = 5
710         elif event.key() == Qt.Key.Key_6:
711             self.yBarGraph = np.array([sum(self.ydata[5][i:i+self.av_height])//self.av_height
     for i in range(0,len(self.ydata[5]),self.av_height)])
712             self.channel = 6
713         elif event.key() == Qt.Key.Key_7:
714             self.yBarGraph = np.array([sum(self.ydata[6][i:i+self.av_height])//self.av_height
     for i in range(0,len(self.ydata[6]),self.av_height)])
715             self.channel = 7
716         elif event.key() == Qt.Key.Key_8:
717             self.yBarGraph = np.array([sum(self.ydata[7][i:i+self.av_height])//self.av_height
     for i in range(0,len(self.ydata[7]),self.av_height)])
718             self.channel = 8
719 
720 
721         # to simulate the accuracy plot
722         if event.key() == Qt.Key_P:
723             self.accuracy_data = np.append(self.accuracy_data, random.sample(range(int(self.
    accuracy_data[-1]), 100), 1))
724             self.accuracy_data_iter = np.append(self.accuracy_data_iter, self.
    accuracy_data_iter[-1] + 1)
725             self.loss_data = np.append(self.loss_data, random.sample(range(0, self.loss_data
    [-1] + 1), 1))
726             self.loss_data_iter = np.append(self.loss_data_iter, self.loss_data_iter[-1] + 1)
727             self.update_ML_plots()
728 
729         self.update()
730 
731 #============================================================================
732 # User Window class
```

```
#=====================================================================
class UserWindow(QMainWindow):
    def __init__(self):
        super(UserWindow, self).__init__()
        self.ui = Ui_UserWindow()
        self.ui.setupUi(self) # Import UI from ui_userWindow.py

        self.setWindowTitle("User Window")

        # Timer
        self.timer = QTimer()
        self.promptTimer = QTimer()

        self.stepsize = 10
        # Check clicked buttons and call their respective functions
        self.timer.timeout.connect(lambda: self.changePages())
        self.promptTimer.timeout.connect(lambda: self.changePrompt())

        # Cap connection
        self.streams = resolve_stream()
        #self.inlet = StreamInlet(self.streams[0])
        recProcess = subprocess.Popen(["python3", "-u", "MeasurementSubgroup/Streaming/
    LSL_csv.py"], stdin=subprocess.PIPE, stdout=subprocess.PIPE,)

    @Slot()
    def startRecording(self):
        global count
        global pageArray
        global i
        i = 0
        count = 0
        pageArray = [1,2,3,4 ,4,3,2,1 ,2,3,4,1 ,1,3,4,2 ,3,2,4,1 ,4,1,2,3, 0] # Fixed prompts
     for labeling
        recProcess.stdout.read1(1)
        recProcess.stdin.write(b"G\n") # G for go
        recProcess.stdin.flush()
        self.timer.start(6000)

    def changePages(self):
        if self.ui.demosPages.currentWidget() == self.ui.trainingPage:
            global count
            global pageArray
            global i

            pageNumber = pageArray[i]

            recProcess.stdin.write(b"Prompt\n") # G for go
            recProcess.stdin.flush()

            if count % 2 != 0:
                self.ui.promptsWidgets.setCurrentWidget(self.ui.calibrationPage)
            else:
                self.ui.promptsWidgets.setCurrentIndex(pageNumber)
                i = i + 1
            if count == 47:
                recProcess.stdin.write(b"Done\n") # G for go
                recProcess.stdin.flush()
                self.timer.stop()

            count = count + 1

    @Slot()
    def stopRecording(self):
        recProcess.stdin.write(b"Stop\n") # G for go
        recProcess.stdin.flush()
        self.timer.stop()
        self.ui.promptsWidgets.setCurrentWidget(self.ui.calibrationPage)

    @Slot()
    def startPromptTimer(self):
            self.promptTimer.start(5000) # Countdown to show prompt
```

```
802
803     def changePrompt(self):
804             self.ui.promptTestWidget.setCurrentWidget(self.ui.promptPromptPage)
805             self.promptTimer.stop()
806
807     # Change pages according to signal received from MainWindow
808     @Slot()
809     def handle_signal_cursorPage(self):
810         self.ui.demosPages.setCurrentWidget(self.ui.cursorPage)
811     @Slot()
812     def handle_signal_trainingPage(self):
813         self.ui.demosPages.setCurrentWidget(self.ui.trainingPage)
814     @Slot()
815     def handle_signal_promptPage(self):
816         self.ui.demosPages.setCurrentWidget(self.ui.promptPage)
817         self.ui.promptTestWidget.setCurrentWidget(self.ui.calibrationPromptPage)
818         self.promptTimer.stop()
819
820     # Simulate cursor movements with WASD keys
821     def keyPressEvent(self, event):
822         if event.key() == Qt.Key.Key_W:
823             if self.ui.mouseCursor.y() - self.stepsize > 0:
824                 self.ui.mouseCursor.move(self.ui.mouseCursor.x(), self.ui.mouseCursor.y() -
    self.stepsize)
825         elif event.key() == Qt.Key.Key_A:
826             if self.ui.mouseCursor.x() - self.stepsize > 0:
827                 self.ui.mouseCursor.move(self.ui.mouseCursor.x() - self.stepsize, self.ui.
    mouseCursor.y())
828         elif event.key() == Qt.Key.Key_S:
829             if self.ui.mouseCursor.y() + self.stepsize < (self.ui.cursorFrame.height() - self
    .ui.mouseCursor.height()):
830                 self.ui.mouseCursor.move(self.ui.mouseCursor.x(), self.ui.mouseCursor.y() +
    self.stepsize)
831         elif event.key() == Qt.Key.Key_D:
832             if self.ui.mouseCursor.x() + self.stepsize < (self.ui.cursorFrame.width() - self.
    ui.mouseCursor.width()):
833                 self.ui.mouseCursor.move(self.ui.mouseCursor.x() + self.stepsize, self.ui.
    mouseCursor.y())
834
835 #============================================================================
836 # ERDS window class
837 #============================================================================
838 class ERDSWindow(QMainWindow):
839
840     def __init__(self):
841         super(ERDSWindow, self).__init__()
842         self.ui = Ui_ERDSWindow()
843         self.ui.setupUi(self) # Import UI from ui_ERDSWindow.py
844
845         self.setWindowTitle("ERDS Window")
846
847 #============================================================================
848 # Lava Game window class
849 #============================================================================
850 class LavaGame(QMainWindow):
851     def __init__(self):
852         super().__init__()
853         self.setWindowTitle("The Floor is Lava")
854         self.central_widget = QWidget(self)
855         self.setCentralWidget(self.central_widget)
856
857         self.grid_layout = QGridLayout(self.central_widget)
858         self.central_widget.setLayout(self.grid_layout)
859
860         self.create_grid()
861         self.create_player()
862         # Timer to check collision with red tiles
863         self.check_collision_timer = QTimer(self)
864         self.check_collision_timer.timeout.connect(self.check_collision)
865
866         self.game_over = False
```

```
867
868         # Countdown setup
869         self.countdown_label = QLabel(self.central_widget)
870         self.countdown_label.setAlignment(Qt.AlignCenter)
871         self.countdown_label.setStyleSheet("font-size: 100px; color: red;")
872         self.countdown_label.setGeometry(0, 0, self.width(), self.height())
873         self.countdown_timer = QTimer(self)
874         self.countdown_timer.timeout.connect(self.update_countdown)
875         self.countdown_value = 5
876
877         # Start the game with countdown
878         self.start_countdown()
879
880     def start_countdown(self):
881         if self.isHidden():
882             pass
883         else:
884             self.countdown_value = 5  # Start countdown from 5
885             self.countdown_label.setText(str(self.countdown_value))
886             self.countdown_label.show()
887             self.countdown_timer.start(1000)
888
889     def update_countdown(self):
890         self.countdown_value -= 1
891         if self.countdown_value > 0:
892             self.countdown_label.setText(str(self.countdown_value))
893         else:
894             self.countdown_timer.stop()
895             self.countdown_label.hide()
896             self.generate_warning()
897
898     def start_game(self):
899         if self.isHidden():
900             pass
901         else:
902             QTimer.singleShot(3000, self.generate_warning)  # Schedule turning new warning
    tiles after 3 seconds
903
904     def create_grid(self):
905         grid_size = 8
906         self.red_tiles = set()
907         self.empty_cells = set()  # Store the empty cells separately
908
909         # Add empty cells to the left
910         for row in range(grid_size):
911             empty_cell_widget = QWidget()
912             empty_cell_widget.setStyleSheet("border: none;")  # No border or background color
913             self.grid_layout.addWidget(empty_cell_widget, row, 0)
914             self.empty_cells.add(empty_cell_widget)  # Add empty cells to the set
915
916         for row in range(grid_size):
917             empty_cell_widget = QWidget()
918             empty_cell_widget.setStyleSheet("border: none;")  # No border or background color
919             self.grid_layout.addWidget(empty_cell_widget, row, 1)
920             self.empty_cells.add(empty_cell_widget)  # Add empty cells to the set
921
922         # Add empty cells to the right
923         for row in range(grid_size):
924             empty_cell_widget = QWidget()
925             empty_cell_widget.setStyleSheet("border: none;")  # No border or background color
926             self.grid_layout.addWidget(empty_cell_widget, row, grid_size + 1)
927             self.empty_cells.add(empty_cell_widget)  # Add empty cells to the set
928
929         for row in range(grid_size):
930             empty_cell_widget = QWidget()
931             empty_cell_widget.setStyleSheet("border: none;")  # No border or background color
932             self.grid_layout.addWidget(empty_cell_widget, row, grid_size + 2)
933             self.empty_cells.add(empty_cell_widget)  # Add empty cells to the set
934
935         for row in range(grid_size):
936             empty_cell_widget = QWidget()
```

```
937             empty_cell_widget.setStyleSheet("border: none;")  # No border or background color
938             self.grid_layout.addWidget(empty_cell_widget, row, grid_size + 3)
939             self.empty_cells.add(empty_cell_widget)  # Add empty cells to the set
940 
941         # Add game cells
942         for row in range(grid_size):
943             for col in range(grid_size):
944                 cell_widget = QWidget()
945                 cell_widget.setStyleSheet("background-color: white; border: 1px solid black;"
    )
946                 self.grid_layout.addWidget(cell_widget, row, col + 2)  # Offset by 2 to skip
    the empty columns
947 
948     def create_player(self):
949         self.player = QWidget(self.central_widget)
950         self.player.setFixedSize(120, 120)
951         self.player.setStyleSheet("background-color: blue; border: 1px solid black;")
952         self.player.move(1220, 640)  # Position the player initially
953 
954     def generate_warning(self):
955         if self.isHidden():
956             pass
957         else:
958             self.check_collision_timer.start(50)  # Check collision every 50 milliseconds
959             if self.game_over:
960                 return
961             for tile in self.red_tiles:
962                 tile.setStyleSheet("background-color: white; border: 1px solid black;")
963             self.red_tiles.clear()
964 
965             num_warning_tiles = random.randint(4, 6)
966             available_cells = [self.grid_layout.itemAt(i).widget() for i in range(self.
    grid_layout.count()) if self.grid_layout.itemAt(i).widget() not in self.empty_cells]
967             warning_tiles = random.sample(available_cells, num_warning_tiles)
968             for tile in warning_tiles:
969                 tile.setStyleSheet("background-color: yellow; border: 1px solid black;")
970                 self.red_tiles.add(tile)
971 
972             QTimer.singleShot(6000, self.generate_lava)  # Schedule turning warning tiles to
    lava after 6 seconds
973 
974     def generate_lava(self):
975         if self.isHidden():
976             pass
977         else:
978             for tile in self.red_tiles:
979                 tile.setStyleSheet("background-color: red; border: 1px solid black;")
980             QTimer.singleShot(3000, self.revert_lava)  # Schedule reverting lava tiles to
    white after 3 seconds
981 
982     def revert_lava(self):
983         for tile in self.red_tiles:
984             tile.setStyleSheet("background-color: white; border: 1px solid black;")
985         self.start_game()  # Start a new cycle of the game
986 
987     # Check if there is a collision between player and lava tile
988     def check_collision(self):
989         if not self.game_over:
990             player_rect = self.player.geometry()
991             for tile in self.red_tiles:
992                 if tile.styleSheet() == "background-color: red; border: 1px solid black;" and
     player_rect.intersects(tile.geometry()):
993                     self.game_over = True
994                     self.game_over_label = QLabel("Game Over", self.central_widget)
995                     self.game_over_label.setAlignment(Qt.AlignmentFlag.AlignCenter)
996                     self.game_over_label.setGeometry(0, 0, self.width(), self.height())
997                     self.game_over_label.setStyleSheet("font-size: 100px; color: red;")
998                     self.game_over_label.raise_()
999                     self.game_over_label.show()
1000 
1001     def keyPressEvent(self, event):
```

```
1002         if not self.game_over:
1003             self.step = 40  # Define step size for movement
1004             if event.key() == Qt.Key.Key_W:
1005                 if self.player.y() - self.step + 10 > 0:
1006                     self.player.move(self.player.x(), self.player.y() - self.step)
1007             elif event.key() == Qt.Key.Key_A:
1008                 if self.player.x() - self.step > self.width()/6:
1009                     self.player.move(self.player.x() - self.step, self.player.y())
1010             elif event.key() == Qt.Key.Key_S:
1011                 if self.player.y() + self.step < (self.height() - self.player.height()):
1012                     self.player.move(self.player.x(), self.player.y() + self.step)
1013             elif event.key() == Qt.Key.Key_D:
1014                 if self.player.x() + self.step < (self.width() - self.player.width())-self.
    width()/6:
1015                     self.player.move(self.player.x() + self.step, self.player.y())
1016 
1017     def closeEvent(self, event):
1018         # Stop all game timers and reset game state
1019         self.check_collision_timer.stop()
1020         self.countdown_timer.stop()
1021         self.game_over = False  # Reset game over flag
1022         self.countdown_label.hide()
1023         if hasattr(self, 'game_over_label'):
1024             self.game_over_label.hide()
1025         event.accept()  # Accept the close event
1026 
1027     def showEvent(self, event):
1028         # Restart the game when the window is shown
1029         self.start_countdown()
1030         event.accept()  # Accept the show event
1031 
1032 #=============================================================================
1033 # Asteroid Game window classes
1034 #=============================================================================
1035 class Asteroid(QMainWindow):
1036     def __init__(self):
1037         super(Asteroid, self).__init__()
1038 
1039         # creating a board object
1040         self.game = Game(self)
1041 
1042         # adding board as a central widget
1043         self.setCentralWidget(self.game)
1044 
1045         # setting title to the window
1046         self.setWindowTitle('Asteroid Shooter')
1047 
1048         # setting geometry to the window
1049         self.setGeometry(100, 100, 600, 400)
1050 
1051         # starting the board object
1052         self.game.start()
1053 
1054 #=============================================================================
1055 # Asteroid Game Frame classes
1056 #=============================================================================
1057 class Game(QFrame):
1058     # timer countdown time
1059     SPEED = 80
1060 
1061     # meteor settings
1062     MAXMETEORS = 3
1063     METEOR_SPEED = 3
1064 
1065     # constructor
1066     def __init__(self, parent):
1067         super(Game, self).__init__(parent)
1068 
1069         # creating a timer
1070         self.timer = QBasicTimer()
1071 
```

```
1072        # player location
1073        self.playerloc = 750
1074
1075        # meteor list
1076        self.meteor = []
1077        # bullet list
1078        self.bullet = []
1079
1080        self.spawn_bullet = False
1081
1082        # keeps track of score
1083        self.score = 0
1084        self.lives = 3
1085
1086        # sizes of objects
1087        self.playerSize = 70
1088        self.cometSize = 70
1089        self.border_size = 470
1090
1091        # direction of player
1092        self.direction = -1
1093
1094        self.game_active = True
1095
1096        # setting focus
1097        self.setFocusPolicy(Qt.FocusPolicy.StrongFocus)
1098
1099    # start method
1100    def start(self):
1101        # starting timer
1102        self.timer.start(Game.SPEED, self)
1103
1104    # paint event
1105    def paintEvent(self, event):
1106        painter = QPainter(self)
1107
1108        # draw side panels
1109        self.draw_side_panels(painter)
1110
1111        # draw the meteors
1112        for pos in self.meteor:
1113            self.draw_square(painter, pos[0], pos[1], self.cometSize, self.cometSize, QColor
    (0xFF0000))
1114
1115        # draw the bullets
1116        for pos in self.bullet:
1117            self.draw_square(painter, pos[0], pos[1], 5, 10, QColor(0xFF0000))
1118
1119        # draw the score (if the game is active, not active when player is gameOver)
1120        if self.game_active:
1121            self.draw_score(painter)
1122
1123            # drawing player
1124            self.draw_square(painter, self.playerloc, self.height() - self.playerSize,
1125                             self.playerSize, self.playerSize, QColor(0x228B22))
1126        else:
1127            self.draw_game_over(painter)
1128
1129    # drawing side panels
1130    def draw_side_panels(self, painter):
1131        # left panel
1132        painter.fillRect(0, 0, self.border_size, self.height(), QColor(0x404040))
1133        # right panel
1134        painter.fillRect(self.width() - self.border_size, 0, self.border_size, self.height(),
     QColor(0x404040))
1135
1136    # drawing score
1137    def draw_score(self, painter):
1138        painter.setPen(QColor(0xFFFFFF))
1139        painter.setFont(QFont('Arial', 50))
1140        score_text = f"Score: {self.score}"
```

```
1141            painter.drawText(10, 70, score_text)
1142            painter.setFont(QFont('Arial', 40))
1143            lives_text = f"Lives: {self.lives}"
1144            painter.drawText(10, 120, lives_text)
1145
1146        # drawing square
1147        def draw_square(self, painter, x, y, width, height, color):
1148            painter.fillRect(x, y, width, height, color)
1149
1150        # draw Game Over text with score
1151        def draw_game_over(self, painter):
1152            painter.setPen(QColor(0xFF0000))
1153            painter.setFont(QFont('Arial', 50))
1154            game_over_text = f"GAME OVER"
1155            text_width = painter.fontMetrics().horizontalAdvance(game_over_text)
1156            painter.drawText((self.width() - text_width) // 2, self.height() // 2, game_over_text
    )
1157            painter.setPen(QColor(0x000000))
1158            painter.setFont(QFont('Arial', 30))
1159            score_text = f"Score: {self.score}"
1160            text_width = painter.fontMetrics().horizontalAdvance(score_text)
1161            painter.drawText((self.width() - text_width) // 2, self.height() // 2 + 50,
    score_text)
1162
1163        # key press event
1164        def keyPressEvent(self, event):
1165            key = event.key()
1166            # if left key is pressed
1167            if key == Qt.Key.Key_A:
1168                # if direction is not right
1169                self.direction = 0
1170
1171            # if right key is pressed
1172            elif key == Qt.Key.Key_D:
1173                # if direction is not left
1174                self.direction = 1
1175
1176            # if space key is pressed
1177            elif key == Qt.Key.Key_Space:
1178                self.spawn_bullet = True
1179
1180        # method to move the player
1181        def move_player(self):
1182            # if direction is left
1183            if self.direction == 0:
1184                if self.playerloc > self.border_size + self.playerSize // 2:
1185                    self.playerloc -= self.playerSize
1186                # reset direction until new move button is pressed
1187                self.direction = -1
1188            # if direction is right
1189            elif self.direction == 1:
1190                if self.playerloc < self.width() - self.border_size - self.playerSize * 2:
1191                    self.playerloc += self.playerSize
1192                # reset direction until new move button is pressed
1193                self.direction = -1
1194
1195
1196        # time event method
1197        def timerEvent(self, event):
1198            if self.width() < 500:
1199                pass
1200            else:
1201                # checking timer id
1202                if event.timerId() == self.timer.timerId():
1203                    # if the player is not gameover
1204                    if self.game_active:
1205                        # move the player and spawn meteors and bullets if needed
1206                        self.move_player()
1207                        self.spawn_meteor()
1208                        if self.spawn_bullet:
```

```
1209                        self.bullet.append([self.playerloc + self.playerSize // 2, self.
    height() - self.playerSize])
1210                        self.spawn_bullet = False
1211                    # call update meteor and bullet methods
1212                    self.update_meteor()
1213                    self.update_bullet()
1214                    # update the window
1215                    self.update()
1216
1217    # spawns a meteor
1218    def spawn_meteor(self):
1219        # if there are less than the max amount of meteors, spawn one
1220        if len(self.meteor) < self.MAXMETEORS:
1221            # getting new random x location until its not the same as an already existing
    meteor's location
1222            while True:
1223                # creating random x coord for the meteor within vertical field
1224                x = random.randint(self.border_size, self.width() - self.border_size - self.
    cometSize)
1225                # extract x-coordinates of existing meteors and check for overlap
1226                overlapping = False
1227                for pos in self.meteor:
1228                    if abs(x - pos[0]) < self.cometSize:
1229                        overlapping = True
1230                        break
1231                if not overlapping:
1232                    break
1233            self.meteor.append([x, 0])
1234
1235    # move the meteors down
1236    def update_meteor(self):
1237        for index, pos in enumerate(self.meteor[:]):
1238            pos[1] += Game.METEOR_SPEED
1239            # if it hits the ground, remove a life and the meteor
1240            if pos[1] > self.height() - self.playerSize:
1241                self.lives -= 1
1242                self.meteor.remove(pos)
1243                # call Game over method when no lives are left
1244                if self.lives <= 0:
1245                    self.game_over()
1246
1247    # when no lives are left, destroy all bullets and meteors
1248    def game_over(self):
1249        self.game_active = False
1250        for index, pos in enumerate(self.meteor[:]):
1251            self.meteor.remove(pos)
1252        for index, pos in enumerate(self.meteor[:]):
1253            self.bullet.remove(pos)
1254
1255    # move all bullets up and check for collision
1256    def update_bullet(self):
1257        for pos in self.bullet[:]:
1258            pos[1] -= self.playerSize
1259            # if bullet is too high, destroy it
1260            if pos[1] < 0:
1261                self.bullet.remove(pos)
1262            # if bullet collides with meteor, remove the bullet and meteor and add a point to
     the players score
1263            for pos_meteor in self.meteor[:]:
1264                if pos[0] > pos_meteor[0] and pos[0] < pos_meteor[0] + self.cometSize and pos
    [1] < pos_meteor[1] + self.cometSize:
1265                    self.bullet.remove(pos)
1266                    self.meteor.remove(pos_meteor)
1267                    self.score += 1
1268
1269 def show_main_window():
1270     window1.showMaximized()
1271     window1.show()
1272
1273     splash.finish(window1)
1274
```

```
#Creates the app and runs the Mainwindow
if __name__ == "__main__":

    path = './users.csv'
    if not os.path.isfile(path):
        with open('users.csv', 'w') as file:
            writer = csv.writer(file)
            writer.writerow(["Name", "ID"])

    app = QApplication(sys.argv)


    splash = SplashScreen()
    splash.show()

    window1 = MainWindow()
    window2 = UserWindow()
    window3 = ERDSWindow()
    window4 = LavaGame()
    window5 = Asteroid()

    window1.setUserWindow(window2)
    window1.setERDSWindow(window3)
    window1.setLavaGameWindow(window4)
    window1.setAsteroidWindow(window5)

    window1.userWindow_to_cursorPage.connect(window2.handle_signal_cursorPage)
    window1.userWindow_to_promptPage.connect(window2.handle_signal_promptPage)
    window1.userWindow_to_trainingPage.connect(window2.handle_signal_trainingPage)
    window1.userWindow_startRecording.connect(window2.startRecording)
    window1.userWindow_stopRecording.connect(window2.stopRecording)
    window1.userWindow_startPromptTimer.connect(window2.startPromptTimer)

    QTimer.singleShot(3000, show_main_window)

    sys.exit(app.exec_())
```

9.2. UI FILES

9.2.1. UI_INTERFACE.PY

```
# -*- coding: utf-8 -*-

################################################################################
## Form generated from reading UI file 'interfacetaijxH.ui'
##
## Created by: Qt User Interface Compiler version 6.4.3
##
## WARNING! All changes made in this file will be lost when recompiling UI file!
################################################################################

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QAbstractItemView, QApplication, QFrame, QGridLayout,
    QHBoxLayout, QLabel, QLineEdit, QListWidget,
    QListWidgetItem, QMainWindow, QPushButton, QSizePolicy,
    QSpacerItem, QVBoxLayout, QWidget)

from Custom_Widgets.QCustomQStackedWidget import QCustomQStackedWidget
from Custom_Widgets.QCustomSlideMenu import QCustomSlideMenu
from pyqtgraph import (GraphicsLayoutWidget, PlotWidget)
import image_rc
import resources_rc

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
```

```
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(1595, 862)
        MainWindow.setMinimumSize(QSize(0, 0))
        icon = QIcon()
        icon.addFile(u":/newPrefix/pictures/EEG.jpg", QSize(), QIcon.Normal, QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setStyleSheet(u"*{\n"
" background-color:transparent;\n"
" background:none;\n"
" padding:0;\n"
" margin:0;\n"
" color:#000;\n"
"}\n"
"\n"
"")
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.horizontalLayout = QHBoxLayout(self.centralwidget)
        self.horizontalLayout.setSpacing(0)
        self.horizontalLayout.setObjectName(u"horizontalLayout")
        self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
        self.leftMenuContainer = QCustomSlideMenu(self.centralwidget)
        self.leftMenuContainer.setObjectName(u"leftMenuContainer")
        self.leftMenuContainer.setMaximumSize(QSize(39, 16777215))
        self.leftMenuContainer.setStyleSheet(u"*{\n"
"background-color: rgb(0, 166, 214);\n"
"border:none;\n"
"color: rgb(255, 255, 255);\n"
"}\n"
"QPushButton{\n"
" text-align:left;\n"
"}")
        self.verticalLayout = QVBoxLayout(self.leftMenuContainer)
        self.verticalLayout.setSpacing(0)
        self.verticalLayout.setObjectName(u"verticalLayout")
        self.verticalLayout.setContentsMargins(5, 0, 0, 0)
        self.leftSubMenuContainer = QWidget(self.leftMenuContainer)
        self.leftSubMenuContainer.setObjectName(u"leftSubMenuContainer")
        self.verticalLayout_2 = QVBoxLayout(self.leftSubMenuContainer)
        self.verticalLayout_2.setSpacing(0)
        self.verticalLayout_2.setObjectName(u"verticalLayout_2")
        self.verticalLayout_2.setContentsMargins(0, 0, 0, 0)
        self.frame = QFrame(self.leftSubMenuContainer)
        self.frame.setObjectName(u"frame")
        self.frame.setFrameShape(QFrame.StyledPanel)
        self.frame.setFrameShadow(QFrame.Raised)
        self.horizontalLayout_2 = QHBoxLayout(self.frame)
        self.horizontalLayout_2.setSpacing(0)
        self.horizontalLayout_2.setObjectName(u"horizontalLayout_2")
        self.horizontalLayout_2.setContentsMargins(0, 0, 0, 0)
        self.menuBtn = QPushButton(self.frame)
        self.menuBtn.setObjectName(u"menuBtn")
        font = QFont()
        font.setPointSize(22)
        font.setBold(True)
        self.menuBtn.setFont(font)
        icon1 = QIcon()
        icon1.addFile(u":/icons/Qss/icons/menu.svg", QSize(), QIcon.Normal, QIcon.Off)
        self.menuBtn.setIcon(icon1)
        self.menuBtn.setIconSize(QSize(30, 30))

        self.horizontalLayout_2.addWidget(self.menuBtn)


        self.verticalLayout_2.addWidget(self.frame)

        self.frame_2 = QFrame(self.leftSubMenuContainer)
        self.frame_2.setObjectName(u"frame_2")
        self.frame_2.setStyleSheet(u"border-radius: 10px;")
        self.frame_2.setFrameShape(QFrame.StyledPanel)
```

```
102         self.frame_2.setFrameShadow(QFrame.Raised)
103         self.verticalLayout_4 = QVBoxLayout(self.frame_2)
104         self.verticalLayout_4.setSpacing(0)
105         self.verticalLayout_4.setObjectName(u"verticalLayout_4")
106         self.verticalLayout_4.setContentsMargins(0, 20, 0, 0)
107         self.overviewBtn = QPushButton(self.frame_2)
108         self.overviewBtn.setObjectName(u"overviewBtn")
109         font1 = QFont()
110         font1.setPointSize(22)
111         self.overviewBtn.setFont(font1)
112         self.overviewBtn.setStyleSheet(u"")
113         icon2 = QIcon()
114         icon2.addFile(u":/icons/Qss/icons/activity-svgrepo-com.svg", QSize(), QIcon.Normal,
    QIcon.Off)
115         self.overviewBtn.setIcon(icon2)
116         self.overviewBtn.setIconSize(QSize(30, 30))
117 
118         self.verticalLayout_4.addWidget(self.overviewBtn)
119 
120         self.demosBtn = QPushButton(self.frame_2)
121         self.demosBtn.setObjectName(u"demosBtn")
122         self.demosBtn.setFont(font1)
123         icon3 = QIcon()
124         icon3.addFile(u":/icons/Qss/icons/console-controller-gamepad-play-svgrepo-com.svg",
    QSize(), QIcon.Normal, QIcon.Off)
125         self.demosBtn.setIcon(icon3)
126         self.demosBtn.setIconSize(QSize(30, 30))
127 
128         self.verticalLayout_4.addWidget(self.demosBtn)
129 
130         self.usersBtn = QPushButton(self.frame_2)
131         self.usersBtn.setObjectName(u"usersBtn")
132         self.usersBtn.setFont(font1)
133         self.usersBtn.setStyleSheet(u"background-color: rgb(0, 118, 194);")
134         icon4 = QIcon()
135         icon4.addFile(u":/icons/Qss/icons/users.svg", QSize(), QIcon.Normal, QIcon.Off)
136         self.usersBtn.setIcon(icon4)
137         self.usersBtn.setIconSize(QSize(30, 30))
138 
139         self.verticalLayout_4.addWidget(self.usersBtn)
140 
141 
142         self.verticalLayout_2.addWidget(self.frame_2, 0, Qt.AlignTop)
143 
144         self.verticalSpacer = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
145 
146         self.verticalLayout_2.addItem(self.verticalSpacer)
147 
148         self.frame_3 = QFrame(self.leftSubMenuContainer)
149         self.frame_3.setObjectName(u"frame_3")
150         self.frame_3.setFrameShape(QFrame.StyledPanel)
151         self.frame_3.setFrameShadow(QFrame.Raised)
152         self.verticalLayout_5 = QVBoxLayout(self.frame_3)
153         self.verticalLayout_5.setSpacing(0)
154         self.verticalLayout_5.setObjectName(u"verticalLayout_5")
155         self.verticalLayout_5.setContentsMargins(0, 10, 0, 10)
156         self.reconnectBtn = QPushButton(self.frame_3)
157         self.reconnectBtn.setObjectName(u"reconnectBtn")
158         self.reconnectBtn.setFont(font1)
159         icon5 = QIcon()
160         icon5.addFile(u":/icons/Qss/icons/download-cloud.svg", QSize(), QIcon.Normal, QIcon.
    Off)
161         self.reconnectBtn.setIcon(icon5)
162         self.reconnectBtn.setIconSize(QSize(30, 30))
163 
164         self.verticalLayout_5.addWidget(self.reconnectBtn)
165 
166         self.infoBtn = QPushButton(self.frame_3)
167         self.infoBtn.setObjectName(u"infoBtn")
168         self.infoBtn.setFont(font1)
169         icon6 = QIcon()
```

```
170         icon6.addFile(u":/icons/Qss/icons/info.svg", QSize(), QIcon.Normal, QIcon.Off)
171         self.infoBtn.setIcon(icon6)
172         self.infoBtn.setIconSize(QSize(30, 30))
173 
174         self.verticalLayout_5.addWidget(self.infoBtn)
175 
176         self.exitBtn = QPushButton(self.frame_3)
177         self.exitBtn.setObjectName(u"exitBtn")
178         self.exitBtn.setFont(font1)
179         icon7 = QIcon()
180         icon7.addFile(u":/icons/Qss/icons/x-circle.svg", QSize(), QIcon.Normal, QIcon.Off)
181         self.exitBtn.setIcon(icon7)
182         self.exitBtn.setIconSize(QSize(30, 30))
183 
184         self.verticalLayout_5.addWidget(self.exitBtn)
185 
186 
187         self.verticalLayout_2.addWidget(self.frame_3)
188 
189 
190         self.verticalLayout.addWidget(self.leftSubMenuContainer)
191 
192 
193         self.horizontalLayout.addWidget(self.leftMenuContainer, 0, Qt.AlignLeft)
194 
195         self.leftSubMenu = QCustomSlideMenu(self.centralwidget)
196         self.leftSubMenu.setObjectName(u"leftSubMenu")
197         self.leftSubMenu.setStyleSheet(u"*{\n"
198 "background-color: rgb(0, 118, 194);\n"
199 "border:none;\n"
200 "color: rgb(255, 255, 255);\n"
201 "}\n"
202 "QPushButton{\n"
203 " text-align:left;\n"
204 "}")
205         self.verticalLayout_7 = QVBoxLayout(self.leftSubMenu)
206         self.verticalLayout_7.setObjectName(u"verticalLayout_7")
207         self.cursorBtn = QPushButton(self.leftSubMenu)
208         self.cursorBtn.setObjectName(u"cursorBtn")
209         self.cursorBtn.setFont(font1)
210         icon8 = QIcon()
211         icon8.addFile(u":/icons/Qss/icons/cursor-square-svgrepo-com.svg", QSize(), QIcon.
    Normal, QIcon.Off)
212         self.cursorBtn.setIcon(icon8)
213         self.cursorBtn.setIconSize(QSize(30, 30))
214 
215         self.verticalLayout_7.addWidget(self.cursorBtn)
216 
217         self.trainBtn = QPushButton(self.leftSubMenu)
218         self.trainBtn.setObjectName(u"trainBtn")
219         self.trainBtn.setFont(font1)
220         icon9 = QIcon()
221         icon9.addFile(u":/icons/Qss/icons/target.svg", QSize(), QIcon.Normal, QIcon.Off)
222         self.trainBtn.setIcon(icon9)
223         self.trainBtn.setIconSize(QSize(30, 30))
224 
225         self.verticalLayout_7.addWidget(self.trainBtn)
226 
227         self.game1Btn = QPushButton(self.leftSubMenu)
228         self.game1Btn.setObjectName(u"game1Btn")
229         self.game1Btn.setFont(font1)
230         icon10 = QIcon()
231         icon10.addFile(u":/icons/Qss/icons/fire-svgrepo-com.svg", QSize(), QIcon.Normal,
    QIcon.Off)
232         self.game1Btn.setIcon(icon10)
233         self.game1Btn.setIconSize(QSize(30, 30))
234 
235         self.verticalLayout_7.addWidget(self.game1Btn)
236 
237         self.game2Btn = QPushButton(self.leftSubMenu)
238         self.game2Btn.setObjectName(u"game2Btn")
```

```
239         self.game2Btn.setFont(font1)
240         icon11 = QIcon()
241         icon11.addFile(u":/icons/Qss/icons/alien-gray-line-drawing-svgrepo-com.svg", QSize(),
     QIcon.Normal, QIcon.Off)
242         self.game2Btn.setIcon(icon11)
243         self.game2Btn.setIconSize(QSize(30, 30))
244 
245         self.verticalLayout_7.addWidget(self.game2Btn)
246 
247         self.verticalSpacer_3 = QSpacerItem(20, 631, QSizePolicy.Minimum, QSizePolicy.
    Expanding)
248 
249         self.verticalLayout_7.addItem(self.verticalSpacer_3)
250 
251 
252         self.horizontalLayout.addWidget(self.leftSubMenu)
253 
254         self.mainPages = QCustomQStackedWidget(self.centralwidget)
255         self.mainPages.setObjectName(u"mainPages")
256         sizePolicy = QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
257         sizePolicy.setHorizontalStretch(0)
258         sizePolicy.setVerticalStretch(0)
259         sizePolicy.setHeightForWidth(self.mainPages.sizePolicy().hasHeightForWidth())
260         self.mainPages.setSizePolicy(sizePolicy)
261         self.mainPages.setStyleSheet(u"background-color: rgb(255, 255, 255);")
262         self.mainBodyContainerGUI = QWidget()
263         self.mainBodyContainerGUI.setObjectName(u"mainBodyContainerGUI")
264         self.mainBodyContainerGUI.setStyleSheet(u"")
265         self.gridLayout = QGridLayout(self.mainBodyContainerGUI)
266         self.gridLayout.setSpacing(6)
267         self.gridLayout.setObjectName(u"gridLayout")
268         self.gridLayout.setContentsMargins(-1, -1, -1, 9)
269         self.rightBodyFrameOverview = QWidget(self.mainBodyContainerGUI)
270         self.rightBodyFrameOverview.setObjectName(u"rightBodyFrameOverview")
271         sizePolicy1 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
272         sizePolicy1.setHorizontalStretch(50)
273         sizePolicy1.setVerticalStretch(95)
274         sizePolicy1.setHeightForWidth(self.rightBodyFrameOverview.sizePolicy().
    hasHeightForWidth())
275         self.rightBodyFrameOverview.setSizePolicy(sizePolicy1)
276         self.rightBodyFrameOverview.setStyleSheet(u"border:none;")
277         self.verticalLayout_6 = QVBoxLayout(self.rightBodyFrameOverview)
278         self.verticalLayout_6.setSpacing(9)
279         self.verticalLayout_6.setObjectName(u"verticalLayout_6")
280         self.verticalLayout_6.setContentsMargins(0, 0, 0, 0)
281         self.buttonsBox = QWidget(self.rightBodyFrameOverview)
282         self.buttonsBox.setObjectName(u"buttonsBox")
283         sizePolicy2 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Fixed)
284         sizePolicy2.setHorizontalStretch(0)
285         sizePolicy2.setVerticalStretch(0)
286         sizePolicy2.setHeightForWidth(self.buttonsBox.sizePolicy().hasHeightForWidth())
287         self.buttonsBox.setSizePolicy(sizePolicy2)
288         self.buttonsBox.setMinimumSize(QSize(600, 114))
289         self.buttonsBox.setStyleSheet(u"*{\n"
290 "border:none;\n"
291 "color: rgb(255, 255, 255);\n"
292 "border-radius: 10px;\n"
293 "background-color: rgb(0, 118, 194);\n"
294 "}\n"
295 "QPushButton{\n"
296 " text-align:center;\n"
297 " background-color: rgb(0, 166, 214);\n"
298 "}")
299         self.gridLayout_3 = QGridLayout(self.buttonsBox)
300         self.gridLayout_3.setObjectName(u"gridLayout_3")
301         self.label_6 = QLabel(self.buttonsBox)
302         self.label_6.setObjectName(u"label_6")
303         font2 = QFont()
304         font2.setPointSize(16)
305         font2.setBold(True)
306         self.label_6.setFont(font2)
```

```
307
308         self.gridLayout_3.addWidget(self.label_6, 0, 0, 1, 1)
309
310         self.openUserWindowBtn = QPushButton(self.buttonsBox)
311         self.openUserWindowBtn.setObjectName(u"openUserWindowBtn")
312         sizePolicy3 = QSizePolicy(QSizePolicy.Minimum, QSizePolicy.Fixed)
313         sizePolicy3.setHorizontalStretch(1)
314         sizePolicy3.setVerticalStretch(0)
315         sizePolicy3.setHeightForWidth(self.openUserWindowBtn.sizePolicy().hasHeightForWidth()
    )
316         self.openUserWindowBtn.setSizePolicy(sizePolicy3)
317         font3 = QFont()
318         font3.setPointSize(16)
319         self.openUserWindowBtn.setFont(font3)
320
321         self.gridLayout_3.addWidget(self.openUserWindowBtn, 1, 0, 1, 1)
322
323         self.openPromptBtn = QPushButton(self.buttonsBox)
324         self.openPromptBtn.setObjectName(u"openPromptBtn")
325         sizePolicy3.setHeightForWidth(self.openPromptBtn.sizePolicy().hasHeightForWidth())
326         self.openPromptBtn.setSizePolicy(sizePolicy3)
327         self.openPromptBtn.setFont(font3)
328
329         self.gridLayout_3.addWidget(self.openPromptBtn, 1, 1, 1, 1)
330
331         self.startTimerBtn = QPushButton(self.buttonsBox)
332         self.startTimerBtn.setObjectName(u"startTimerBtn")
333         sizePolicy4 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Fixed)
334         sizePolicy4.setHorizontalStretch(1)
335         sizePolicy4.setVerticalStretch(0)
336         sizePolicy4.setHeightForWidth(self.startTimerBtn.sizePolicy().hasHeightForWidth())
337         self.startTimerBtn.setSizePolicy(sizePolicy4)
338         self.startTimerBtn.setFont(font3)
339
340         self.gridLayout_3.addWidget(self.startTimerBtn, 1, 2, 1, 1)
341
342         self.promptTimeFrame = QFrame(self.buttonsBox)
343         self.promptTimeFrame.setObjectName(u"promptTimeFrame")
344         sizePolicy2.setHeightForWidth(self.promptTimeFrame.sizePolicy().hasHeightForWidth())
345         self.promptTimeFrame.setSizePolicy(sizePolicy2)
346         self.promptTimeFrame.setStyleSheet(u"background-color: rgb(0, 166, 214);")
347         self.promptTimeFrame.setFrameShape(QFrame.StyledPanel)
348         self.promptTimeFrame.setFrameShadow(QFrame.Raised)
349         self.horizontalLayout_4 = QHBoxLayout(self.promptTimeFrame)
350         self.horizontalLayout_4.setSpacing(0)
351         self.horizontalLayout_4.setObjectName(u"horizontalLayout_4")
352         self.horizontalLayout_4.setContentsMargins(0, 0, 0, 0)
353         self.stopwatch = QLabel(self.promptTimeFrame)
354         self.stopwatch.setObjectName(u"stopwatch")
355         sizePolicy5 = QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Preferred)
356         sizePolicy5.setHorizontalStretch(75)
357         sizePolicy5.setVerticalStretch(0)
358         sizePolicy5.setHeightForWidth(self.stopwatch.sizePolicy().hasHeightForWidth())
359         self.stopwatch.setSizePolicy(sizePolicy5)
360         self.stopwatch.setFont(font3)
361         self.stopwatch.setAlignment(Qt.AlignCenter)
362
363         self.horizontalLayout_4.addWidget(self.stopwatch)
364
365         self.unit_sec = QLabel(self.promptTimeFrame)
366         self.unit_sec.setObjectName(u"unit_sec")
367         sizePolicy6 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
368         sizePolicy6.setHorizontalStretch(25)
369         sizePolicy6.setVerticalStretch(0)
370         sizePolicy6.setHeightForWidth(self.unit_sec.sizePolicy().hasHeightForWidth())
371         self.unit_sec.setSizePolicy(sizePolicy6)
372         self.unit_sec.setFont(font3)
373
374         self.horizontalLayout_4.addWidget(self.unit_sec)
375
376
```

```
377         self.gridLayout_3.addWidget(self.promptTimeFrame, 1, 3, 1, 1)
378
379         self.dataTrainingBtn = QPushButton(self.buttonsBox)
380         self.dataTrainingBtn.setObjectName(u"dataTrainingBtn")
381         sizePolicy4.setHeightForWidth(self.dataTrainingBtn.sizePolicy().hasHeightForWidth())
382         self.dataTrainingBtn.setSizePolicy(sizePolicy4)
383         self.dataTrainingBtn.setFont(font3)
384
385         self.gridLayout_3.addWidget(self.dataTrainingBtn, 2, 0, 1, 1)
386
387         self.startRecordingBtn = QPushButton(self.buttonsBox)
388         self.startRecordingBtn.setObjectName(u"startRecordingBtn")
389         sizePolicy4.setHeightForWidth(self.startRecordingBtn.sizePolicy().hasHeightForWidth()
    )
390         self.startRecordingBtn.setSizePolicy(sizePolicy4)
391         self.startRecordingBtn.setFont(font3)
392
393         self.gridLayout_3.addWidget(self.startRecordingBtn, 2, 1, 1, 1)
394
395         self.stopRecordingBtn = QPushButton(self.buttonsBox)
396         self.stopRecordingBtn.setObjectName(u"stopRecordingBtn")
397         sizePolicy4.setHeightForWidth(self.stopRecordingBtn.sizePolicy().hasHeightForWidth())
398         self.stopRecordingBtn.setSizePolicy(sizePolicy4)
399         self.stopRecordingBtn.setFont(font3)
400
401         self.gridLayout_3.addWidget(self.stopRecordingBtn, 2, 2, 1, 1)
402
403         self.ERDSBtn = QPushButton(self.buttonsBox)
404         self.ERDSBtn.setObjectName(u"ERDSBtn")
405         sizePolicy4.setHeightForWidth(self.ERDSBtn.sizePolicy().hasHeightForWidth())
406         self.ERDSBtn.setSizePolicy(sizePolicy4)
407         self.ERDSBtn.setFont(font3)
408
409         self.gridLayout_3.addWidget(self.ERDSBtn, 2, 3, 1, 1)
410
411
412         self.verticalLayout_6.addWidget(self.buttonsBox)
413
414         self.FFTFrame = QFrame(self.rightBodyFrameOverview)
415         self.FFTFrame.setObjectName(u"FFTFrame")
416         sizePolicy7 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
417         sizePolicy7.setHorizontalStretch(0)
418         sizePolicy7.setVerticalStretch(40)
419         sizePolicy7.setHeightForWidth(self.FFTFrame.sizePolicy().hasHeightForWidth())
420         self.FFTFrame.setSizePolicy(sizePolicy7)
421         self.FFTFrame.setStyleSheet(u"")
422         self.FFTFrame.setFrameShape(QFrame.StyledPanel)
423         self.FFTFrame.setFrameShadow(QFrame.Raised)
424         self.horizontalLayout_6 = QHBoxLayout(self.FFTFrame)
425         self.horizontalLayout_6.setSpacing(0)
426         self.horizontalLayout_6.setObjectName(u"horizontalLayout_6")
427         self.horizontalLayout_6.setContentsMargins(0, 0, 0, 0)
428         self.FFTPlot = PlotWidget(self.FFTFrame)
429         self.FFTPlot.setObjectName(u"FFTPlot")
430         brush = QBrush(QColor(255, 255, 255, 255))
431         brush.setStyle(Qt.NoBrush)
432         self.FFTPlot.setBackgroundBrush(brush)
433
434         self.horizontalLayout_6.addWidget(self.FFTPlot)
435
436
437         self.verticalLayout_6.addWidget(self.FFTFrame)
438
439         self.powerBandFrame = QFrame(self.rightBodyFrameOverview)
440         self.powerBandFrame.setObjectName(u"powerBandFrame")
441         sizePolicy7.setHeightForWidth(self.powerBandFrame.sizePolicy().hasHeightForWidth())
442         self.powerBandFrame.setSizePolicy(sizePolicy7)
443         self.powerBandFrame.setFrameShape(QFrame.StyledPanel)
444         self.powerBandFrame.setFrameShadow(QFrame.Raised)
445         self.horizontalLayout_10 = QHBoxLayout(self.powerBandFrame)
446         self.horizontalLayout_10.setObjectName(u"horizontalLayout_10")
```

```
447         self.horizontalLayout_10.setContentsMargins(0, 0, 0, 0)
448         self.powerBandPlot = PlotWidget(self.powerBandFrame)
449         self.powerBandPlot.setObjectName(u"powerBandPlot")
450         brush1 = QBrush(QColor(255, 255, 255, 255))
451         brush1.setStyle(Qt.NoBrush)
452         self.powerBandPlot.setBackgroundBrush(brush1)
453 
454         self.horizontalLayout_10.addWidget(self.powerBandPlot)
455 
456 
457         self.verticalLayout_6.addWidget(self.powerBandFrame)
458 
459 
460         self.gridLayout.addWidget(self.rightBodyFrameOverview, 1, 1, 1, 1)
461 
462         self.UserIDBox = QWidget(self.mainBodyContainerGUI)
463         self.UserIDBox.setObjectName(u"UserIDBox")
464         sizePolicy2.setHeightForWidth(self.UserIDBox.sizePolicy().hasHeightForWidth())
465         self.UserIDBox.setSizePolicy(sizePolicy2)
466         self.UserIDBox.setMaximumSize(QSize(16777215, 70))
467         self.UserIDBox.setStyleSheet(u"color: rgb(255, 255, 255);\n"
468 "background-color: rgb(0, 166, 214);\n"
469 "border-radius: 10px;")
470         self.horizontalLayout_15 = QHBoxLayout(self.UserIDBox)
471         self.horizontalLayout_15.setSpacing(0)
472         self.horizontalLayout_15.setObjectName(u"horizontalLayout_15")
473         self.horizontalLayout_15.setContentsMargins(0, 0, 100, 0)
474         self.logoFrame = QFrame(self.UserIDBox)
475         self.logoFrame.setObjectName(u"logoFrame")
476         sizePolicy8 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
477         sizePolicy8.setHorizontalStretch(10)
478         sizePolicy8.setVerticalStretch(0)
479         sizePolicy8.setHeightForWidth(self.logoFrame.sizePolicy().hasHeightForWidth())
480         self.logoFrame.setSizePolicy(sizePolicy8)
481         self.logoFrame.setStyleSheet(u"image: url(:/newPrefix/pictures/TUDelft-logo_white.png
    );")
482         self.logoFrame.setFrameShape(QFrame.StyledPanel)
483         self.logoFrame.setFrameShadow(QFrame.Raised)
484 
485         self.horizontalLayout_15.addWidget(self.logoFrame)
486 
487         self.userID_test = QLineEdit(self.UserIDBox)
488         self.userID_test.setObjectName(u"userID_test")
489         sizePolicy9 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Fixed)
490         sizePolicy9.setHorizontalStretch(90)
491         sizePolicy9.setVerticalStretch(0)
492         sizePolicy9.setHeightForWidth(self.userID_test.sizePolicy().hasHeightForWidth())
493         self.userID_test.setSizePolicy(sizePolicy9)
494         font4 = QFont()
495         font4.setPointSize(28)
496         font4.setBold(True)
497         self.userID_test.setFont(font4)
498         self.userID_test.setAlignment(Qt.AlignCenter)
499 
500         self.horizontalLayout_15.addWidget(self.userID_test)
501 
502 
503         self.gridLayout.addWidget(self.UserIDBox, 0, 0, 1, 2)
504 
505         self.leftBodyFrameOverview = QWidget(self.mainBodyContainerGUI)
506         self.leftBodyFrameOverview.setObjectName(u"leftBodyFrameOverview")
507         sizePolicy10 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Expanding)
508         sizePolicy10.setHorizontalStretch(50)
509         sizePolicy10.setVerticalStretch(95)
510         sizePolicy10.setHeightForWidth(self.leftBodyFrameOverview.sizePolicy().
    hasHeightForWidth())
511         self.leftBodyFrameOverview.setSizePolicy(sizePolicy10)
512         self.leftBodyFrameOverview.setStyleSheet(u"background-color: rgb(255, 255, 255);")
513         self.verticalLayout_3 = QVBoxLayout(self.leftBodyFrameOverview)
514         self.verticalLayout_3.setSpacing(6)
515         self.verticalLayout_3.setObjectName(u"verticalLayout_3")
```

```
516         self.verticalLayout_3.setContentsMargins(0, 0, 0, 0)
517         self.directionFrame = QFrame(self.leftBodyFrameOverview)
518         self.directionFrame.setObjectName(u"directionFrame")
519         sizePolicy11 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Minimum)
520         sizePolicy11.setHorizontalStretch(0)
521         sizePolicy11.setVerticalStretch(10)
522         sizePolicy11.setHeightForWidth(self.directionFrame.sizePolicy().hasHeightForWidth())
523         self.directionFrame.setSizePolicy(sizePolicy11)
524         self.directionFrame.setStyleSheet(u"background-color: rgb(0, 0, 0);\n"
525 "border-radius: 10px")
526         self.directionFrame.setFrameShape(QFrame.StyledPanel)
527         self.directionFrame.setFrameShadow(QFrame.Raised)
528         self.horizontalLayout_3 = QHBoxLayout(self.directionFrame)
529         self.horizontalLayout_3.setSpacing(3)
530         self.horizontalLayout_3.setObjectName(u"horizontalLayout_3")
531         self.horizontalLayout_3.setContentsMargins(3, 3, 3, 3)
532         self.directionLine = QLineEdit(self.directionFrame)
533         self.directionLine.setObjectName(u"directionLine")
534         sizePolicy12 = QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
535         sizePolicy12.setHorizontalStretch(50)
536         sizePolicy12.setVerticalStretch(0)
537         sizePolicy12.setHeightForWidth(self.directionLine.sizePolicy().hasHeightForWidth())
538         self.directionLine.setSizePolicy(sizePolicy12)
539         font5 = QFont()
540         font5.setPointSize(24)
541         self.directionLine.setFont(font5)
542         self.directionLine.setCursor(QCursor(Qt.ArrowCursor))
543         self.directionLine.setStyleSheet(u"background-color: rgb(255, 255, 255);")
544         self.directionLine.setAlignment(Qt.AlignCenter)
545         self.directionLine.setReadOnly(True)
546 
547         self.horizontalLayout_3.addWidget(self.directionLine)
548 
549 
550         self.verticalLayout_3.addWidget(self.directionFrame)
551 
552         self.valuesFrame = QFrame(self.leftBodyFrameOverview)
553         self.valuesFrame.setObjectName(u"valuesFrame")
554         sizePolicy13 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
555         sizePolicy13.setHorizontalStretch(0)
556         sizePolicy13.setVerticalStretch(15)
557         sizePolicy13.setHeightForWidth(self.valuesFrame.sizePolicy().hasHeightForWidth())
558         self.valuesFrame.setSizePolicy(sizePolicy13)
559         self.valuesFrame.setMaximumSize(QSize(16777215, 107))
560         self.valuesFrame.setStyleSheet(u"background-color: rgb(0, 118, 194);\n"
561 "color: rgb(255, 255, 255);\n"
562 "border-radius: 10px;")
563         self.valuesFrame.setFrameShape(QFrame.StyledPanel)
564         self.valuesFrame.setFrameShadow(QFrame.Raised)
565         self.gridLayout_2 = QGridLayout(self.valuesFrame)
566         self.gridLayout_2.setSpacing(0)
567         self.gridLayout_2.setObjectName(u"gridLayout_2")
568         self.gridLayout_2.setContentsMargins(9, 9, 9, 9)
569         self.valuesBox = QHBoxLayout()
570         self.valuesBox.setObjectName(u"valuesBox")
571         self.verticalLayout_8 = QVBoxLayout()
572         self.verticalLayout_8.setObjectName(u"verticalLayout_8")
573         self.label = QLabel(self.valuesFrame)
574         self.label.setObjectName(u"label")
575         sizePolicy14 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Expanding)
576         sizePolicy14.setHorizontalStretch(0)
577         sizePolicy14.setVerticalStretch(0)
578         sizePolicy14.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
579         self.label.setSizePolicy(sizePolicy14)
580         self.label.setFont(font3)
581         self.label.setStyleSheet(u"border:none;")
582         self.label.setAlignment(Qt.AlignRight|Qt.AlignTrailing|Qt.AlignVCenter)
583 
584         self.verticalLayout_8.addWidget(self.label)
585 
586         self.label_2 = QLabel(self.valuesFrame)
```

```
587         self.label_2.setObjectName(u"label_2")
588         sizePolicy14.setHeightForWidth(self.label_2.sizePolicy().hasHeightForWidth())
589         self.label_2.setSizePolicy(sizePolicy14)
590         self.label_2.setFont(font3)
591         self.label_2.setStyleSheet(u"border:none;")
592         self.label_2.setAlignment(Qt.AlignRight|Qt.AlignTrailing|Qt.AlignVCenter)
593 
594         self.verticalLayout_8.addWidget(self.label_2)
595 
596 
597         self.valuesBox.addLayout(self.verticalLayout_8)
598 
599         self.verticalLayout_9 = QVBoxLayout()
600         self.verticalLayout_9.setObjectName(u"verticalLayout_9")
601         self.learningRateLine = QLineEdit(self.valuesFrame)
602         self.learningRateLine.setObjectName(u"learningRateLine")
603         sizePolicy.setHeightForWidth(self.learningRateLine.sizePolicy().hasHeightForWidth())
604         self.learningRateLine.setSizePolicy(sizePolicy)
605         self.learningRateLine.setFont(font3)
606         self.learningRateLine.setCursor(QCursor(Qt.ArrowCursor))
607         self.learningRateLine.setStyleSheet(u"")
608 
609         self.verticalLayout_9.addWidget(self.learningRateLine)
610 
611         self.batchSizeLine = QLineEdit(self.valuesFrame)
612         self.batchSizeLine.setObjectName(u"batchSizeLine")
613         sizePolicy.setHeightForWidth(self.batchSizeLine.sizePolicy().hasHeightForWidth())
614         self.batchSizeLine.setSizePolicy(sizePolicy)
615         self.batchSizeLine.setFont(font3)
616         self.batchSizeLine.setCursor(QCursor(Qt.ArrowCursor))
617 
618         self.verticalLayout_9.addWidget(self.batchSizeLine)
619 
620 
621         self.valuesBox.addLayout(self.verticalLayout_9)
622 
623         self.verticalLayout_10 = QVBoxLayout()
624         self.verticalLayout_10.setObjectName(u"verticalLayout_10")
625         self.label_3 = QLabel(self.valuesFrame)
626         self.label_3.setObjectName(u"label_3")
627         sizePolicy14.setHeightForWidth(self.label_3.sizePolicy().hasHeightForWidth())
628         self.label_3.setSizePolicy(sizePolicy14)
629         self.label_3.setFont(font3)
630         self.label_3.setStyleSheet(u"border:none;")
631         self.label_3.setAlignment(Qt.AlignRight|Qt.AlignTrailing|Qt.AlignVCenter)
632 
633         self.verticalLayout_10.addWidget(self.label_3)
634 
635         self.label_4 = QLabel(self.valuesFrame)
636         self.label_4.setObjectName(u"label_4")
637         sizePolicy14.setHeightForWidth(self.label_4.sizePolicy().hasHeightForWidth())
638         self.label_4.setSizePolicy(sizePolicy14)
639         self.label_4.setFont(font3)
640         self.label_4.setStyleSheet(u"border:none;")
641         self.label_4.setAlignment(Qt.AlignRight|Qt.AlignTrailing|Qt.AlignVCenter)
642 
643         self.verticalLayout_10.addWidget(self.label_4)
644 
645 
646         self.valuesBox.addLayout(self.verticalLayout_10)
647 
648         self.verticalLayout_11 = QVBoxLayout()
649         self.verticalLayout_11.setObjectName(u"verticalLayout_11")
650         self.maxIterationLine = QLineEdit(self.valuesFrame)
651         self.maxIterationLine.setObjectName(u"maxIterationLine")
652         sizePolicy.setHeightForWidth(self.maxIterationLine.sizePolicy().hasHeightForWidth())
653         self.maxIterationLine.setSizePolicy(sizePolicy)
654         self.maxIterationLine.setFont(font3)
655         self.maxIterationLine.setCursor(QCursor(Qt.ArrowCursor))
656 
657         self.verticalLayout_11.addWidget(self.maxIterationLine)
```

```
658
659         self.marginLine = QLineEdit(self.valuesFrame)
660         self.marginLine.setObjectName(u"marginLine")
661         sizePolicy.setHeightForWidth(self.marginLine.sizePolicy().hasHeightForWidth())
662         self.marginLine.setSizePolicy(sizePolicy)
663         self.marginLine.setFont(font3)
664         self.marginLine.setCursor(QCursor(Qt.ArrowCursor))
665
666         self.verticalLayout_11.addWidget(self.marginLine)
667
668
669         self.valuesBox.addLayout(self.verticalLayout_11)
670
671
672         self.gridLayout_2.addLayout(self.valuesBox, 1, 0, 1, 1)
673
674         self.label_5 = QLabel(self.valuesFrame)
675         self.label_5.setObjectName(u"label_5")
676         self.label_5.setFont(font2)
677
678         self.gridLayout_2.addWidget(self.label_5, 0, 0, 1, 1)
679
680
681         self.verticalLayout_3.addWidget(self.valuesFrame)
682
683         self.channelsFrame = QFrame(self.leftBodyFrameOverview)
684         self.channelsFrame.setObjectName(u"channelsFrame")
685         sizePolicy15 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
686         sizePolicy15.setHorizontalStretch(0)
687         sizePolicy15.setVerticalStretch(60)
688         sizePolicy15.setHeightForWidth(self.channelsFrame.sizePolicy().hasHeightForWidth())
689         self.channelsFrame.setSizePolicy(sizePolicy15)
690         self.channelsFrame.setSizeIncrement(QSize(0, 0))
691         self.channelsFrame.setBaseSize(QSize(0, 0))
692         self.channelsFrame.setStyleSheet(u"")
693         self.channelsFrame.setFrameShape(QFrame.StyledPanel)
694         self.channelsFrame.setFrameShadow(QFrame.Raised)
695         self.horizontalLayout_9 = QHBoxLayout(self.channelsFrame)
696         self.horizontalLayout_9.setObjectName(u"horizontalLayout_9")
697         self.horizontalLayout_8 = QHBoxLayout()
698         self.horizontalLayout_8.setObjectName(u"horizontalLayout_8")
699         self.verticalLayout_14 = QVBoxLayout()
700         self.verticalLayout_14.setObjectName(u"verticalLayout_14")
701         self.ch1Label = QLabel(self.channelsFrame)
702         self.ch1Label.setObjectName(u"ch1Label")
703         sizePolicy8.setHeightForWidth(self.ch1Label.sizePolicy().hasHeightForWidth())
704         self.ch1Label.setSizePolicy(sizePolicy8)
705         self.ch1Label.setFont(font3)
706         self.ch1Label.setAlignment(Qt.AlignCenter)
707
708         self.verticalLayout_14.addWidget(self.ch1Label)
709
710         self.ch2Label = QLabel(self.channelsFrame)
711         self.ch2Label.setObjectName(u"ch2Label")
712         sizePolicy8.setHeightForWidth(self.ch2Label.sizePolicy().hasHeightForWidth())
713         self.ch2Label.setSizePolicy(sizePolicy8)
714         self.ch2Label.setFont(font3)
715         self.ch2Label.setAlignment(Qt.AlignCenter)
716
717         self.verticalLayout_14.addWidget(self.ch2Label)
718
719         self.ch3Label = QLabel(self.channelsFrame)
720         self.ch3Label.setObjectName(u"ch3Label")
721         sizePolicy8.setHeightForWidth(self.ch3Label.sizePolicy().hasHeightForWidth())
722         self.ch3Label.setSizePolicy(sizePolicy8)
723         self.ch3Label.setFont(font3)
724         self.ch3Label.setAlignment(Qt.AlignCenter)
725
726         self.verticalLayout_14.addWidget(self.ch3Label)
727
728         self.ch4Label = QLabel(self.channelsFrame)
```

```
729         self.ch4Label.setObjectName(u"ch4Label")
730         sizePolicy8.setHeightForWidth(self.ch4Label.sizePolicy().hasHeightForWidth())
731         self.ch4Label.setSizePolicy(sizePolicy8)
732         self.ch4Label.setFont(font3)
733         self.ch4Label.setAlignment(Qt.AlignCenter)
734 
735         self.verticalLayout_14.addWidget(self.ch4Label)
736 
737         self.ch5Label = QLabel(self.channelsFrame)
738         self.ch5Label.setObjectName(u"ch5Label")
739         sizePolicy8.setHeightForWidth(self.ch5Label.sizePolicy().hasHeightForWidth())
740         self.ch5Label.setSizePolicy(sizePolicy8)
741         self.ch5Label.setFont(font3)
742         self.ch5Label.setAlignment(Qt.AlignCenter)
743 
744         self.verticalLayout_14.addWidget(self.ch5Label)
745 
746         self.ch6Label = QLabel(self.channelsFrame)
747         self.ch6Label.setObjectName(u"ch6Label")
748         sizePolicy8.setHeightForWidth(self.ch6Label.sizePolicy().hasHeightForWidth())
749         self.ch6Label.setSizePolicy(sizePolicy8)
750         self.ch6Label.setFont(font3)
751         self.ch6Label.setAlignment(Qt.AlignCenter)
752 
753         self.verticalLayout_14.addWidget(self.ch6Label)
754 
755         self.ch7Label = QLabel(self.channelsFrame)
756         self.ch7Label.setObjectName(u"ch7Label")
757         sizePolicy8.setHeightForWidth(self.ch7Label.sizePolicy().hasHeightForWidth())
758         self.ch7Label.setSizePolicy(sizePolicy8)
759         self.ch7Label.setFont(font3)
760         self.ch7Label.setAlignment(Qt.AlignCenter)
761 
762         self.verticalLayout_14.addWidget(self.ch7Label)
763 
764         self.ch8Label = QLabel(self.channelsFrame)
765         self.ch8Label.setObjectName(u"ch8Label")
766         sizePolicy8.setHeightForWidth(self.ch8Label.sizePolicy().hasHeightForWidth())
767         self.ch8Label.setSizePolicy(sizePolicy8)
768         self.ch8Label.setFont(font3)
769         self.ch8Label.setStyleSheet(u"")
770         self.ch8Label.setAlignment(Qt.AlignCenter)
771 
772         self.verticalLayout_14.addWidget(self.ch8Label)
773 
774 
775         self.horizontalLayout_8.addLayout(self.verticalLayout_14)
776 
777         self.widget = QWidget(self.channelsFrame)
778         self.widget.setObjectName(u"widget")
779 
780         self.horizontalLayout_8.addWidget(self.widget)
781 
782         self.channelsPlot = GraphicsLayoutWidget(self.channelsFrame)
783         self.channelsPlot.setObjectName(u"channelsPlot")
784         self.channelsPlot.setStyleSheet(u"background-color: rgb(255, 255, 255);")
785 
786         self.horizontalLayout_8.addWidget(self.channelsPlot)
787 
788 
789         self.horizontalLayout_9.addLayout(self.horizontalLayout_8)
790 
791 
792         self.verticalLayout_3.addWidget(self.channelsFrame)
793 
794 
795         self.gridLayout.addWidget(self.leftBodyFrameOverview, 1, 0, 1, 1)
796 
797         self.infoWidgetContainer = QWidget(self.mainBodyContainerGUI)
798         self.infoWidgetContainer.setObjectName(u"infoWidgetContainer")
799         sizePolicy16 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
```

```
800         sizePolicy16.setHorizontalStretch(0)
801         sizePolicy16.setVerticalStretch(0)
802         sizePolicy16.setHeightForWidth(self.infoWidgetContainer.sizePolicy().
        hasHeightForWidth())
803         self.infoWidgetContainer.setSizePolicy(sizePolicy16)
804         self.infoWidgetContainer.setStyleSheet(u"border-radius: 10px;")
805         self.horizontalLayout_7 = QHBoxLayout(self.infoWidgetContainer)
806         self.horizontalLayout_7.setObjectName(u"horizontalLayout_7")
807         self.infoWidgetTrain = QCustomSlideMenu(self.infoWidgetContainer)
808         self.infoWidgetTrain.setObjectName(u"infoWidgetTrain")
809         sizePolicy16.setHeightForWidth(self.infoWidgetTrain.sizePolicy().hasHeightForWidth())
810         self.infoWidgetTrain.setSizePolicy(sizePolicy16)
811         self.infoWidgetTrain.setMinimumSize(QSize(0, 0))
812         self.infoWidgetTrain.setMaximumSize(QSize(0, 0))
813         self.infoWidgetTrain.setStyleSheet(u"background-color: rgb(0, 184, 200);\n"
814 "color: rgb(255, 255, 255);\n"
815 "border-radius: 10px;")
816         self.verticalLayout_12 = QVBoxLayout(self.infoWidgetTrain)
817         self.verticalLayout_12.setSpacing(0)
818         self.verticalLayout_12.setObjectName(u"verticalLayout_12")
819         self.verticalLayout_12.setContentsMargins(9, 9, 9, 9)
820         self.label_13 = QLabel(self.infoWidgetTrain)
821         self.label_13.setObjectName(u"label_13")
822         font6 = QFont()
823         font6.setPointSize(12)
824         font6.setBold(True)
825         self.label_13.setFont(font6)
826 
827         self.verticalLayout_12.addWidget(self.label_13)
828 
829         self.frame_10 = QFrame(self.infoWidgetTrain)
830         self.frame_10.setObjectName(u"frame_10")
831         sizePolicy16.setHeightForWidth(self.frame_10.sizePolicy().hasHeightForWidth())
832         self.frame_10.setSizePolicy(sizePolicy16)
833         self.frame_10.setStyleSheet(u"border-radius: 10px;")
834         self.frame_10.setFrameShape(QFrame.StyledPanel)
835         self.frame_10.setFrameShadow(QFrame.Raised)
836         self.horizontalLayout_5 = QHBoxLayout(self.frame_10)
837         self.horizontalLayout_5.setSpacing(0)
838         self.horizontalLayout_5.setObjectName(u"horizontalLayout_5")
839         self.horizontalLayout_5.setContentsMargins(0, 0, 0, 0)
840         self.label_14 = QLabel(self.frame_10)
841         self.label_14.setObjectName(u"label_14")
842         sizePolicy17 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
843         sizePolicy17.setHorizontalStretch(80)
844         sizePolicy17.setVerticalStretch(0)
845         sizePolicy17.setHeightForWidth(self.label_14.sizePolicy().hasHeightForWidth())
846         self.label_14.setSizePolicy(sizePolicy17)
847         font7 = QFont()
848         font7.setPointSize(12)
849         self.label_14.setFont(font7)
850 
851         self.horizontalLayout_5.addWidget(self.label_14)
852 
853         self.frame_11 = QFrame(self.frame_10)
854         self.frame_11.setObjectName(u"frame_11")
855         sizePolicy16.setHeightForWidth(self.frame_11.sizePolicy().hasHeightForWidth())
856         self.frame_11.setSizePolicy(sizePolicy16)
857         self.frame_11.setMinimumSize(QSize(0, 0))
858         self.frame_11.setStyleSheet(u"border-radius: 10px;")
859         self.frame_11.setFrameShape(QFrame.StyledPanel)
860         self.frame_11.setFrameShadow(QFrame.Raised)
861         self.verticalLayout_13 = QVBoxLayout(self.frame_11)
862         self.verticalLayout_13.setSpacing(0)
863         self.verticalLayout_13.setObjectName(u"verticalLayout_13")
864         self.verticalLayout_13.setContentsMargins(0, 0, 0, 0)
865         self.widget_11 = QWidget(self.frame_11)
866         self.widget_11.setObjectName(u"widget_11")
867         sizePolicy14.setHeightForWidth(self.widget_11.sizePolicy().hasHeightForWidth())
868         self.widget_11.setSizePolicy(sizePolicy14)
869         self.widget_11.setStyleSheet(u"image: url(:/newPrefix/pictures/EEG cap.png);")
```

```
870
871         self.verticalLayout_13.addWidget(self.widget_11)
872
873         self.label_15 = QLabel(self.frame_11)
874         self.label_15.setObjectName(u"label_15")
875         font8 = QFont()
876         font8.setPointSize(10)
877         self.label_15.setFont(font8)
878
879         self.verticalLayout_13.addWidget(self.label_15)
880
881
882         self.horizontalLayout_5.addWidget(self.frame_11)
883
884
885         self.verticalLayout_12.addWidget(self.frame_10)
886
887
888         self.horizontalLayout_7.addWidget(self.infoWidgetTrain)
889
890
891         self.gridLayout.addWidget(self.infoWidgetContainer, 2, 0, 1, 1)
892
893         self.mainPages.addWidget(self.mainBodyContainerGUI)
894         self.mainBodyContainerUsers = QWidget()
895         self.mainBodyContainerUsers.setObjectName(u"mainBodyContainerUsers")
896         self.mainBodyContainerUsers.setStyleSheet(u"QPushButton{text-align:left; padding:5px
    15px;border-radius: 10px;}")
897         self.gridLayout_6 = QGridLayout(self.mainBodyContainerUsers)
898         self.gridLayout_6.setObjectName(u"gridLayout_6")
899         self.gridLayout_6.setContentsMargins(-1, -1, -1, 9)
900         self.listBtns = QVBoxLayout()
901         self.listBtns.setObjectName(u"listBtns")
902         self.verticalSpacer_2 = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.
    Expanding)
903
904         self.listBtns.addItem(self.verticalSpacer_2)
905
906         self.addBtn = QPushButton(self.mainBodyContainerUsers)
907         self.addBtn.setObjectName(u"addBtn")
908         font9 = QFont()
909         font9.setPointSize(14)
910         self.addBtn.setFont(font9)
911         self.addBtn.setStyleSheet(u"background-color: rgb(0, 166, 214);\n"
912 "color: rgb(255, 255, 255);")
913
914         self.listBtns.addWidget(self.addBtn)
915
916         self.editBtn = QPushButton(self.mainBodyContainerUsers)
917         self.editBtn.setObjectName(u"editBtn")
918         self.editBtn.setFont(font9)
919         self.editBtn.setStyleSheet(u"background-color: rgb(0, 166, 214);\n"
920 "color: rgb(255, 255, 255);")
921
922         self.listBtns.addWidget(self.editBtn)
923
924         self.removeBtn = QPushButton(self.mainBodyContainerUsers)
925         self.removeBtn.setObjectName(u"removeBtn")
926         self.removeBtn.setFont(font9)
927         self.removeBtn.setStyleSheet(u"background-color: rgb(0, 166, 214);\n"
928 "color: rgb(255, 255, 255);")
929
930         self.listBtns.addWidget(self.removeBtn)
931
932         self.upBtn = QPushButton(self.mainBodyContainerUsers)
933         self.upBtn.setObjectName(u"upBtn")
934         self.upBtn.setFont(font9)
935         self.upBtn.setStyleSheet(u"background-color: rgb(0, 166, 214);\n"
936 "color: rgb(255, 255, 255);")
937
938         self.listBtns.addWidget(self.upBtn)
```

```
939
940         self.downBtn = QPushButton(self.mainBodyContainerUsers)
941         self.downBtn.setObjectName(u"downBtn")
942         self.downBtn.setFont(font9)
943         self.downBtn.setStyleSheet(u"background-color: rgb(0, 166, 214);\n"
944 "color: rgb(255, 255, 255);")
945
946         self.listBtns.addWidget(self.downBtn)
947
948         self.sortBtn = QPushButton(self.mainBodyContainerUsers)
949         self.sortBtn.setObjectName(u"sortBtn")
950         self.sortBtn.setFont(font9)
951         self.sortBtn.setStyleSheet(u"background-color: rgb(0, 166, 214);\n"
952 "color: rgb(255, 255, 255);")
953
954         self.listBtns.addWidget(self.sortBtn)
955
956
957         self.gridLayout_6.addLayout(self.listBtns, 1, 1, 1, 1)
958
959         self.usersList = QListWidget(self.mainBodyContainerUsers)
960         self.usersList.setObjectName(u"usersList")
961         sizePolicy18 = QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
962         sizePolicy18.setHorizontalStretch(0)
963         sizePolicy18.setVerticalStretch(90)
964         sizePolicy18.setHeightForWidth(self.usersList.sizePolicy().hasHeightForWidth())
965         self.usersList.setSizePolicy(sizePolicy18)
966         font10 = QFont()
967         font10.setPointSize(18)
968         self.usersList.setFont(font10)
969         self.usersList.setAutoFillBackground(False)
970         self.usersList.setStyleSheet(u"background-color: rgb(232, 232, 232);")
971         self.usersList.setAlternatingRowColors(True)
972         self.usersList.setSelectionBehavior(QAbstractItemView.SelectRows)
973         self.usersList.setSelectionRectVisible(True)
974
975         self.gridLayout_6.addWidget(self.usersList, 1, 0, 1, 1)
976
977         self.horizontalSpacer = QSpacerItem(40, 20, QSizePolicy.Minimum, QSizePolicy.Minimum)
978
979         self.gridLayout_6.addItem(self.horizontalSpacer, 0, 1, 1, 1)
980
981         self.usersBox = QWidget(self.mainBodyContainerUsers)
982         self.usersBox.setObjectName(u"usersBox")
983         self.usersBox.setStyleSheet(u"background-color:transparent;")
984         self.horizontalLayout_16 = QHBoxLayout(self.usersBox)
985         self.horizontalLayout_16.setSpacing(0)
986         self.horizontalLayout_16.setObjectName(u"horizontalLayout_16")
987         self.horizontalLayout_16.setContentsMargins(0, 0, 0, 0)
988         self.logo2Frame = QFrame(self.usersBox)
989         self.logo2Frame.setObjectName(u"logo2Frame")
990         sizePolicy8.setHeightForWidth(self.logo2Frame.sizePolicy().hasHeightForWidth())
991         self.logo2Frame.setSizePolicy(sizePolicy8)
992         self.logo2Frame.setStyleSheet(u"image: url(:/newPrefix/pictures/flame logo.png);")
993         self.logo2Frame.setFrameShape(QFrame.StyledPanel)
994         self.logo2Frame.setFrameShadow(QFrame.Raised)
995
996         self.horizontalLayout_16.addWidget(self.logo2Frame)
997
998         self.usersTitle = QLineEdit(self.usersBox)
999         self.usersTitle.setObjectName(u"usersTitle")
1000        sizePolicy9.setHeightForWidth(self.usersTitle.sizePolicy().hasHeightForWidth())
1001        self.usersTitle.setSizePolicy(sizePolicy9)
1002        font11 = QFont()
1003        font11.setPointSize(50)
1004        self.usersTitle.setFont(font11)
1005        self.usersTitle.setStyleSheet(u"border:none;\n"
1006 "color: rgb(12, 35, 64);")
1007        self.usersTitle.setAlignment(Qt.AlignRight|Qt.AlignTrailing|Qt.AlignVCenter)
1008        self.usersTitle.setReadOnly(True)
1009
```

```
1010         self.horizontalLayout_16.addWidget(self.usersTitle)
1011 
1012 
1013         self.gridLayout_6.addWidget(self.usersBox, 0, 0, 1, 1)
1014 
1015         self.mainPages.addWidget(self.mainBodyContainerUsers)
1016 
1017         self.horizontalLayout.addWidget(self.mainPages)
1018 
1019         MainWindow.setCentralWidget(self.centralwidget)
1020 
1021         self.retranslateUi(MainWindow)
1022 
1023         self.mainPages.setCurrentIndex(1)
1024 
1025 
1026         QMetaObject.connectSlotsByName(MainWindow)
1027     # setupUi
1028 
1029     def retranslateUi(self, MainWindow):
1030         MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow", u"MainWindow",
     None))
1031 #if QT_CONFIG(tooltip)
1032         self.menuBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Menu", None))
1033 #endif // QT_CONFIG(tooltip)
1034         self.menuBtn.setText(QCoreApplication.translate("MainWindow", u"Menu", None))
1035 #if QT_CONFIG(tooltip)
1036         self.overviewBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Train", None))
1037 #endif // QT_CONFIG(tooltip)
1038         self.overviewBtn.setText(QCoreApplication.translate("MainWindow", u"Dashboard", None)
     )
1039 #if QT_CONFIG(tooltip)
1040         self.demosBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Demos", None))
1041 #endif // QT_CONFIG(tooltip)
1042         self.demosBtn.setText(QCoreApplication.translate("MainWindow", u"Demos", None))
1043 #if QT_CONFIG(tooltip)
1044         self.usersBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Users", None))
1045 #endif // QT_CONFIG(tooltip)
1046         self.usersBtn.setText(QCoreApplication.translate("MainWindow", u"Users", None))
1047 #if QT_CONFIG(tooltip)
1048         self.reconnectBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Reconnect",
     None))
1049 #endif // QT_CONFIG(tooltip)
1050         self.reconnectBtn.setText(QCoreApplication.translate("MainWindow", u"Reconnect", None
     ))
1051 #if QT_CONFIG(tooltip)
1052         self.infoBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Information", None
     ))
1053 #endif // QT_CONFIG(tooltip)
1054         self.infoBtn.setText(QCoreApplication.translate("MainWindow", u"Info", None))
1055 #if QT_CONFIG(tooltip)
1056         self.exitBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Exit Window", None
     ))
1057 #endif // QT_CONFIG(tooltip)
1058         self.exitBtn.setText(QCoreApplication.translate("MainWindow", u"Exit", None))
1059         self.cursorBtn.setText(QCoreApplication.translate("MainWindow", u"Cursor", None))
1060         self.trainBtn.setText(QCoreApplication.translate("MainWindow", u"Train", None))
1061         self.game1Btn.setText(QCoreApplication.translate("MainWindow", u"Game 1", None))
1062         self.game2Btn.setText(QCoreApplication.translate("MainWindow", u"Game 2", None))
1063         self.label_6.setText(QCoreApplication.translate("MainWindow", u"Control Panel", None)
     )
1064         self.openUserWindowBtn.setText(QCoreApplication.translate("MainWindow", u"Open Window
     ", None))
1065         self.openPromptBtn.setText(QCoreApplication.translate("MainWindow", u"Open Prompt",
     None))
1066         self.startTimerBtn.setText(QCoreApplication.translate("MainWindow", u"Start Timer",
     None))
1067         self.stopwatch.setText(QCoreApplication.translate("MainWindow", u"0", None))
1068         self.unit_sec.setText(QCoreApplication.translate("MainWindow", u"s", None))
1069         self.dataTrainingBtn.setText(QCoreApplication.translate("MainWindow", u"Train Data",
     None))
```

```
1070            self.startRecordingBtn.setText(QCoreApplication.translate("MainWindow", u"Start
    Recording", None))
1071            self.stopRecordingBtn.setText(QCoreApplication.translate("MainWindow", u"Stop
    Recording", None))
1072            self.ERDSBtn.setText(QCoreApplication.translate("MainWindow", u"ERDS", None))
1073            self.userID_test.setText(QCoreApplication.translate("MainWindow", u"No User selected"
    , None))
1074            self.directionLine.setText(QCoreApplication.translate("MainWindow", u"Direction",
    None))
1075            self.label.setText(QCoreApplication.translate("MainWindow", u"Learning rate:   ",
    None))
1076            self.label_2.setText(QCoreApplication.translate("MainWindow", u"Batch size:   ", None
    ))
1077            self.learningRateLine.setText(QCoreApplication.translate("MainWindow", u"100", None))
1078            self.batchSizeLine.setText(QCoreApplication.translate("MainWindow", u"300", None))
1079            self.label_3.setText(QCoreApplication.translate("MainWindow", u"Max iteration:   ",
    None))
1080            self.label_4.setText(QCoreApplication.translate("MainWindow", u"Margin:   ", None))
1081            self.maxIterationLine.setText(QCoreApplication.translate("MainWindow", u"50", None))
1082            self.marginLine.setText(QCoreApplication.translate("MainWindow", u"7", None))
1083            self.label_5.setText(QCoreApplication.translate("MainWindow", u"Training Panel", None
    ))
1084            self.ch1Label.setText(QCoreApplication.translate("MainWindow", u"Ch1", None))
1085            self.ch2Label.setText(QCoreApplication.translate("MainWindow", u"Ch2", None))
1086            self.ch3Label.setText(QCoreApplication.translate("MainWindow", u"Ch3", None))
1087            self.ch4Label.setText(QCoreApplication.translate("MainWindow", u"Ch4", None))
1088            self.ch5Label.setText(QCoreApplication.translate("MainWindow", u"Ch5", None))
1089            self.ch6Label.setText(QCoreApplication.translate("MainWindow", u"Ch6", None))
1090            self.ch7Label.setText(QCoreApplication.translate("MainWindow", u"Ch7", None))
1091            self.ch8Label.setText(QCoreApplication.translate("MainWindow", u"Ch8", None))
1092            self.label_13.setText(QCoreApplication.translate("MainWindow", u"Information", None))
1093            self.label_14.setText(QCoreApplication.translate("MainWindow", u"Welcome to this BCI
    app!\n"
1094 "The dashboard window contains numerous data and parameters for a better understanding of EEG
     data!\n"
1095 " Plots: eight channel plots for each electrode of the cap and an FFT and power band plot\n"
1096 " of the chosen channel. The channel can be changed using the numbers key.\n"
1097 " In the training panel you can change the training values according to your liking.\n"
1098 " Lastly, the buttons panel gives you control over the user's window and other features.\n"
1099 "\n"
1100 " The demos button opens a submenu which contains the possible demo you would like to run.\n"
1101 "\n"
1102 " Lastly, the user's window allows you to choose and edit users that are going to be trained
    or tested.", None))
1103            self.label_15.setText(QCoreApplication.translate("MainWindow", u"Channel numbers with
     according position", None))
1104 #if QT_CONFIG(tooltip)
1105            self.addBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Add user", None))
1106 #endif // QT_CONFIG(tooltip)
1107            self.addBtn.setText(QCoreApplication.translate("MainWindow", u"ADD", None))
1108 #if QT_CONFIG(tooltip)
1109            self.editBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Edit user", None))
1110 #endif // QT_CONFIG(tooltip)
1111            self.editBtn.setText(QCoreApplication.translate("MainWindow", u"EDIT", None))
1112 #if QT_CONFIG(tooltip)
1113            self.removeBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Remove user",
    None))
1114 #endif // QT_CONFIG(tooltip)
1115            self.removeBtn.setText(QCoreApplication.translate("MainWindow", u"REMOVE", None))
1116 #if QT_CONFIG(tooltip)
1117            self.upBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Put user a position
    above", None))
1118 #endif // QT_CONFIG(tooltip)
1119            self.upBtn.setText(QCoreApplication.translate("MainWindow", u"UP", None))
1120 #if QT_CONFIG(tooltip)
1121            self.downBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Put user a
    position above", None))
1122 #endif // QT_CONFIG(tooltip)
1123            self.downBtn.setText(QCoreApplication.translate("MainWindow", u"DOWN", None))
1124 #if QT_CONFIG(tooltip)
1125            self.sortBtn.setToolTip(QCoreApplication.translate("MainWindow", u"Sort users", None)
```

```
    )
#endif // QT_CONFIG(tooltip)
        self.sortBtn.setText(QCoreApplication.translate("MainWindow", u"SORT", None))
        self.usersTitle.setText(QCoreApplication.translate("MainWindow", u"Users", None))
    # retranslateUi
```

9.2.2. ui_userWindow.py

```
# -*- coding: utf-8 -*-

################################################################################
## Form generated from reading UI file 'userWindowuAqIuf.ui'
##
## Created by: Qt User Interface Compiler version 6.4.3
##
## WARNING! All changes made in this file will be lost when recompiling UI file!
################################################################################

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QFrame, QGridLayout, QHBoxLayout,
    QLabel, QMainWindow, QSizePolicy, QSpacerItem,
    QStackedWidget, QWidget)
import image_rc
import resources_rc

class Ui_UserWindow(object):
    def setupUi(self, UserWindow):
        if not UserWindow.objectName():
            UserWindow.setObjectName(u"UserWindow")
        UserWindow.resize(1228, 737)
        icon = QIcon()
        icon.addFile(u":/newPrefix/pictures/EEG.jpg", QSize(), QIcon.Normal, QIcon.Off)
        UserWindow.setWindowIcon(icon)
        UserWindow.setStyleSheet(u"color: rgb(0, 0, 0);\n"
"background-color: rgb(255, 255, 255);")
        self.centralwidget = QWidget(UserWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.horizontalLayout_12 = QHBoxLayout(self.centralwidget)
        self.horizontalLayout_12.setSpacing(0)
        self.horizontalLayout_12.setObjectName(u"horizontalLayout_12")
        self.horizontalLayout_12.setContentsMargins(0, 0, 0, 0)
        self.demosPages = QStackedWidget(self.centralwidget)
        self.demosPages.setObjectName(u"demosPages")
        self.trainingPage = QWidget()
        self.trainingPage.setObjectName(u"trainingPage")
        self.trainingPage.setStyleSheet(u"*{\n"
" background-color:rgb(0, 0, 0);\n"
" color: rgb(200,200,200);\n"
"}")
        self.horizontalLayout = QHBoxLayout(self.trainingPage)
        self.horizontalLayout.setSpacing(0)
        self.horizontalLayout.setObjectName(u"horizontalLayout")
        self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
        self.promptsWidgets = QStackedWidget(self.trainingPage)
        self.promptsWidgets.setObjectName(u"promptsWidgets")
        self.promptsWidgets.setStyleSheet(u"")
        self.calibrationPage = QWidget()
        self.calibrationPage.setObjectName(u"calibrationPage")
        self.calibrationPage.setStyleSheet(u"")
        self.gridLayout = QGridLayout(self.calibrationPage)
        self.gridLayout.setObjectName(u"gridLayout")
        self.verticalSpacer = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.Preferred
    )

        self.gridLayout.addItem(self.verticalSpacer, 0, 1, 1, 1)
```

```
63
64          self.horizontalSpacer_2 = QSpacerItem(348, 20, QSizePolicy.Preferred, QSizePolicy.
        Minimum)
65
66          self.gridLayout.addItem(self.horizontalSpacer_2, 1, 0, 1, 1)
67
68          self.calibrationCross = QFrame(self.calibrationPage)
69          self.calibrationCross.setObjectName(u"calibrationCross")
70          self.calibrationCross.setMaximumSize(QSize(199, 187))
71          self.calibrationCross.setFrameShape(QFrame.StyledPanel)
72          self.calibrationCross.setFrameShadow(QFrame.Raised)
73          self.gridLayout_6 = QGridLayout(self.calibrationCross)
74          self.gridLayout_6.setSpacing(0)
75          self.gridLayout_6.setObjectName(u"gridLayout_6")
76          self.frame_7 = QFrame(self.calibrationCross)
77          self.frame_7.setObjectName(u"frame_7")
78          sizePolicy = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
79          sizePolicy.setHorizontalStretch(0)
80          sizePolicy.setVerticalStretch(0)
81          sizePolicy.setHeightForWidth(self.frame_7.sizePolicy().hasHeightForWidth())
82          self.frame_7.setSizePolicy(sizePolicy)
83          self.frame_7.setMaximumSize(QSize(20, 16777215))
84          self.frame_7.setStyleSheet(u"background-color: rgb(200, 200, 200);")
85          self.frame_7.setFrameShape(QFrame.StyledPanel)
86          self.frame_7.setFrameShadow(QFrame.Raised)
87
88          self.gridLayout_6.addWidget(self.frame_7, 0, 1, 1, 1)
89
90          self.frame_8 = QFrame(self.calibrationCross)
91          self.frame_8.setObjectName(u"frame_8")
92          sizePolicy.setHeightForWidth(self.frame_8.sizePolicy().hasHeightForWidth())
93          self.frame_8.setSizePolicy(sizePolicy)
94          self.frame_8.setStyleSheet(u"background-color: rgb(200, 200, 200);")
95          self.frame_8.setFrameShape(QFrame.StyledPanel)
96          self.frame_8.setFrameShadow(QFrame.Raised)
97
98          self.gridLayout_6.addWidget(self.frame_8, 1, 0, 1, 1)
99
100         self.frame_6 = QFrame(self.calibrationCross)
101         self.frame_6.setObjectName(u"frame_6")
102         sizePolicy.setHeightForWidth(self.frame_6.sizePolicy().hasHeightForWidth())
103         self.frame_6.setSizePolicy(sizePolicy)
104         self.frame_6.setStyleSheet(u"background-color: rgb(200, 200, 200);")
105         self.frame_6.setFrameShape(QFrame.StyledPanel)
106         self.frame_6.setFrameShadow(QFrame.Raised)
107
108         self.gridLayout_6.addWidget(self.frame_6, 1, 1, 1, 1)
109
110         self.frame_9 = QFrame(self.calibrationCross)
111         self.frame_9.setObjectName(u"frame_9")
112         sizePolicy.setHeightForWidth(self.frame_9.sizePolicy().hasHeightForWidth())
113         self.frame_9.setSizePolicy(sizePolicy)
114         self.frame_9.setMaximumSize(QSize(16777215, 20))
115         self.frame_9.setStyleSheet(u"background-color: rgb(200, 200, 200);")
116         self.frame_9.setFrameShape(QFrame.StyledPanel)
117         self.frame_9.setFrameShadow(QFrame.Raised)
118
119         self.gridLayout_6.addWidget(self.frame_9, 1, 2, 1, 1)
120
121         self.frame = QFrame(self.calibrationCross)
122         self.frame.setObjectName(u"frame")
123         sizePolicy.setHeightForWidth(self.frame.sizePolicy().hasHeightForWidth())
124         self.frame.setSizePolicy(sizePolicy)
125         self.frame.setStyleSheet(u"background-color: rgb(200, 200, 200);")
126         self.frame.setFrameShape(QFrame.StyledPanel)
127         self.frame.setFrameShadow(QFrame.Raised)
128
129         self.gridLayout_6.addWidget(self.frame, 2, 1, 1, 1)
130
131
132         self.gridLayout.addWidget(self.calibrationCross, 1, 1, 1, 1)
```

```
133
134         self.horizontalSpacer_3 = QSpacerItem(338, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
135
136         self.gridLayout.addItem(self.horizontalSpacer_3, 1, 2, 1, 1)
137
138         self.verticalSpacer_2 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
139
140         self.gridLayout.addItem(self.verticalSpacer_2, 2, 1, 1, 1)
141
142         self.promptsWidgets.addWidget(self.calibrationPage)
143         self.rightPage = QWidget()
144         self.rightPage.setObjectName(u"rightPage")
145         self.rightPage.setStyleSheet(u"")
146         self.gridLayout_2 = QGridLayout(self.rightPage)
147         self.gridLayout_2.setObjectName(u"gridLayout_2")
148         self.verticalSpacer_3 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
149
150         self.gridLayout_2.addItem(self.verticalSpacer_3, 0, 1, 1, 1)
151
152         self.horizontalSpacer_4 = QSpacerItem(294, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
153
154         self.gridLayout_2.addItem(self.horizontalSpacer_4, 1, 0, 1, 1)
155
156         self.rightFrame = QFrame(self.rightPage)
157         self.rightFrame.setObjectName(u"rightFrame")
158         sizePolicy.setHeightForWidth(self.rightFrame.sizePolicy().hasHeightForWidth())
159         self.rightFrame.setSizePolicy(sizePolicy)
160         self.rightFrame.setStyleSheet(u"")
161         self.rightFrame.setFrameShape(QFrame.StyledPanel)
162         self.rightFrame.setFrameShadow(QFrame.Raised)
163         self.horizontalLayout_2 = QHBoxLayout(self.rightFrame)
164         self.horizontalLayout_2.setObjectName(u"horizontalLayout_2")
165         self.rightLabel = QLabel(self.rightFrame)
166         self.rightLabel.setObjectName(u"rightLabel")
167         font = QFont()
168         font.setPointSize(40)
169         font.setBold(False)
170         self.rightLabel.setFont(font)
171         self.rightLabel.setAlignment(Qt.AlignCenter)
172
173         self.horizontalLayout_2.addWidget(self.rightLabel)
174
175
176         self.gridLayout_2.addWidget(self.rightFrame, 1, 1, 1, 1)
177
178         self.horizontalSpacer_5 = QSpacerItem(294, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
179
180         self.gridLayout_2.addItem(self.horizontalSpacer_5, 1, 2, 1, 1)
181
182         self.verticalSpacer_4 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
183
184         self.gridLayout_2.addItem(self.verticalSpacer_4, 2, 1, 1, 1)
185
186         self.promptsWidgets.addWidget(self.rightPage)
187         self.leftPage = QWidget()
188         self.leftPage.setObjectName(u"leftPage")
189         self.gridLayout_3 = QGridLayout(self.leftPage)
190         self.gridLayout_3.setObjectName(u"gridLayout_3")
191         self.verticalSpacer_5 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
192
193         self.gridLayout_3.addItem(self.verticalSpacer_5, 0, 1, 1, 1)
194
195         self.horizontalSpacer_6 = QSpacerItem(294, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
```

```
196
197         self.gridLayout_3.addItem(self.horizontalSpacer_6, 1, 0, 1, 1)
198
199         self.leftFrame = QFrame(self.leftPage)
200         self.leftFrame.setObjectName(u"leftFrame")
201         sizePolicy.setHeightForWidth(self.leftFrame.sizePolicy().hasHeightForWidth())
202         self.leftFrame.setSizePolicy(sizePolicy)
203         self.leftFrame.setStyleSheet(u"")
204         self.leftFrame.setFrameShape(QFrame.StyledPanel)
205         self.leftFrame.setFrameShadow(QFrame.Raised)
206         self.horizontalLayout_3 = QHBoxLayout(self.leftFrame)
207         self.horizontalLayout_3.setObjectName(u"horizontalLayout_3")
208         self.leftLabel = QLabel(self.leftFrame)
209         self.leftLabel.setObjectName(u"leftLabel")
210         self.leftLabel.setFont(font)
211         self.leftLabel.setAlignment(Qt.AlignCenter)
212
213         self.horizontalLayout_3.addWidget(self.leftLabel)
214
215
216         self.gridLayout_3.addWidget(self.leftFrame, 1, 1, 1, 1)
217
218         self.horizontalSpacer_7 = QSpacerItem(294, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
219
220         self.gridLayout_3.addItem(self.horizontalSpacer_7, 1, 2, 1, 1)
221
222         self.verticalSpacer_6 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
223
224         self.gridLayout_3.addItem(self.verticalSpacer_6, 2, 1, 1, 1)
225
226         self.promptsWidgets.addWidget(self.leftPage)
227         self.upPage = QWidget()
228         self.upPage.setObjectName(u"upPage")
229         self.gridLayout_4 = QGridLayout(self.upPage)
230         self.gridLayout_4.setObjectName(u"gridLayout_4")
231         self.verticalSpacer_7 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
232
233         self.gridLayout_4.addItem(self.verticalSpacer_7, 0, 1, 1, 1)
234
235         self.horizontalSpacer_8 = QSpacerItem(294, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
236
237         self.gridLayout_4.addItem(self.horizontalSpacer_8, 1, 0, 1, 1)
238
239         self.tongueFrame = QFrame(self.upPage)
240         self.tongueFrame.setObjectName(u"tongueFrame")
241         sizePolicy.setHeightForWidth(self.tongueFrame.sizePolicy().hasHeightForWidth())
242         self.tongueFrame.setSizePolicy(sizePolicy)
243         self.tongueFrame.setStyleSheet(u"")
244         self.tongueFrame.setFrameShape(QFrame.StyledPanel)
245         self.tongueFrame.setFrameShadow(QFrame.Raised)
246         self.horizontalLayout_4 = QHBoxLayout(self.tongueFrame)
247         self.horizontalLayout_4.setObjectName(u"horizontalLayout_4")
248         self.tongueLabel = QLabel(self.tongueFrame)
249         self.tongueLabel.setObjectName(u"tongueLabel")
250         font1 = QFont()
251         font1.setPointSize(50)
252         font1.setBold(False)
253         self.tongueLabel.setFont(font1)
254         self.tongueLabel.setAlignment(Qt.AlignCenter)
255
256         self.horizontalLayout_4.addWidget(self.tongueLabel)
257
258
259         self.gridLayout_4.addWidget(self.tongueFrame, 1, 1, 1, 1)
260
261         self.horizontalSpacer_9 = QSpacerItem(294, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
```

```
262
263         self.gridLayout_4.addItem(self.horizontalSpacer_9, 1, 2, 1, 1)
264
265         self.verticalSpacer_8 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
266
267         self.gridLayout_4.addItem(self.verticalSpacer_8, 2, 1, 1, 1)
268
269         self.promptsWidgets.addWidget(self.upPage)
270         self.downPage = QWidget()
271         self.downPage.setObjectName(u"downPage")
272         self.gridLayout_5 = QGridLayout(self.downPage)
273         self.gridLayout_5.setObjectName(u"gridLayout_5")
274         self.verticalSpacer_9 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
275
276         self.gridLayout_5.addItem(self.verticalSpacer_9, 0, 1, 1, 1)
277
278         self.horizontalSpacer_10 = QSpacerItem(294, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
279
280         self.gridLayout_5.addItem(self.horizontalSpacer_10, 1, 0, 1, 1)
281
282         self.feetFrame = QFrame(self.downPage)
283         self.feetFrame.setObjectName(u"feetFrame")
284         sizePolicy.setHeightForWidth(self.feetFrame.sizePolicy().hasHeightForWidth())
285         self.feetFrame.setSizePolicy(sizePolicy)
286         self.feetFrame.setStyleSheet(u"")
287         self.feetFrame.setFrameShape(QFrame.StyledPanel)
288         self.feetFrame.setFrameShadow(QFrame.Raised)
289         self.horizontalLayout_5 = QHBoxLayout(self.feetFrame)
290         self.horizontalLayout_5.setObjectName(u"horizontalLayout_5")
291         self.feetLabel = QLabel(self.feetFrame)
292         self.feetLabel.setObjectName(u"feetLabel")
293         self.feetLabel.setFont(font1)
294         self.feetLabel.setAlignment(Qt.AlignCenter)
295
296         self.horizontalLayout_5.addWidget(self.feetLabel)
297
298
299         self.gridLayout_5.addWidget(self.feetFrame, 1, 1, 1, 1)
300
301         self.horizontalSpacer_11 = QSpacerItem(294, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
302
303         self.gridLayout_5.addItem(self.horizontalSpacer_11, 1, 2, 1, 1)
304
305         self.verticalSpacer_10 = QSpacerItem(20, 185, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
306
307         self.gridLayout_5.addItem(self.verticalSpacer_10, 2, 1, 1, 1)
308
309         self.promptsWidgets.addWidget(self.downPage)
310
311         self.horizontalLayout.addWidget(self.promptsWidgets)
312
313         self.demosPages.addWidget(self.trainingPage)
314         self.promptPage = QWidget()
315         self.promptPage.setObjectName(u"promptPage")
316         self.horizontalLayout_7 = QHBoxLayout(self.promptPage)
317         self.horizontalLayout_7.setSpacing(0)
318         self.horizontalLayout_7.setObjectName(u"horizontalLayout_7")
319         self.horizontalLayout_7.setContentsMargins(0, 0, 0, 0)
320         self.promptTestWidget = QStackedWidget(self.promptPage)
321         self.promptTestWidget.setObjectName(u"promptTestWidget")
322         self.calibrationPromptPage = QWidget()
323         self.calibrationPromptPage.setObjectName(u"calibrationPromptPage")
324         self.calibrationPromptPage.setStyleSheet(u"background-color: rgb(0,0,0);")
325         self.gridLayout_8 = QGridLayout(self.calibrationPromptPage)
326         self.gridLayout_8.setSpacing(0)
327         self.gridLayout_8.setObjectName(u"gridLayout_8")
```

```
328         self.gridLayout_8.setContentsMargins(0, 0, 0, 0)
329         self.frame_2 = QFrame(self.calibrationPromptPage)
330         self.frame_2.setObjectName(u"frame_2")
331         self.frame_2.setFrameShape(QFrame.StyledPanel)
332         self.frame_2.setFrameShadow(QFrame.Raised)
333         self.gridLayout_9 = QGridLayout(self.frame_2)
334         self.gridLayout_9.setObjectName(u"gridLayout_9")
335         self.verticalSpacer_12 = QSpacerItem(20, 247, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
336 
337         self.gridLayout_9.addItem(self.verticalSpacer_12, 0, 1, 1, 1)
338 
339         self.horizontalSpacer_13 = QSpacerItem(487, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
340 
341         self.gridLayout_9.addItem(self.horizontalSpacer_13, 1, 0, 1, 1)
342 
343         self.calibrationCross_2 = QFrame(self.frame_2)
344         self.calibrationCross_2.setObjectName(u"calibrationCross_2")
345         self.calibrationCross_2.setMaximumSize(QSize(199, 187))
346         self.calibrationCross_2.setFrameShape(QFrame.StyledPanel)
347         self.calibrationCross_2.setFrameShadow(QFrame.Raised)
348         self.gridLayout_7 = QGridLayout(self.calibrationCross_2)
349         self.gridLayout_7.setSpacing(0)
350         self.gridLayout_7.setObjectName(u"gridLayout_7")
351         self.frame_11 = QFrame(self.calibrationCross_2)
352         self.frame_11.setObjectName(u"frame_11")
353         sizePolicy.setHeightForWidth(self.frame_11.sizePolicy().hasHeightForWidth())
354         self.frame_11.setSizePolicy(sizePolicy)
355         self.frame_11.setMaximumSize(QSize(20, 16777215))
356         self.frame_11.setStyleSheet(u"background-color: rgb(200, 200, 200);")
357         self.frame_11.setFrameShape(QFrame.StyledPanel)
358         self.frame_11.setFrameShadow(QFrame.Raised)
359 
360         self.gridLayout_7.addWidget(self.frame_11, 0, 1, 1, 1)
361 
362         self.frame_12 = QFrame(self.calibrationCross_2)
363         self.frame_12.setObjectName(u"frame_12")
364         sizePolicy.setHeightForWidth(self.frame_12.sizePolicy().hasHeightForWidth())
365         self.frame_12.setSizePolicy(sizePolicy)
366         self.frame_12.setStyleSheet(u"background-color: rgb(200, 200, 200);")
367         self.frame_12.setFrameShape(QFrame.StyledPanel)
368         self.frame_12.setFrameShadow(QFrame.Raised)
369 
370         self.gridLayout_7.addWidget(self.frame_12, 1, 0, 1, 1)
371 
372         self.frame_13 = QFrame(self.calibrationCross_2)
373         self.frame_13.setObjectName(u"frame_13")
374         sizePolicy.setHeightForWidth(self.frame_13.sizePolicy().hasHeightForWidth())
375         self.frame_13.setSizePolicy(sizePolicy)
376         self.frame_13.setStyleSheet(u"background-color: rgb(200, 200, 200);")
377         self.frame_13.setFrameShape(QFrame.StyledPanel)
378         self.frame_13.setFrameShadow(QFrame.Raised)
379 
380         self.gridLayout_7.addWidget(self.frame_13, 1, 1, 1, 1)
381 
382         self.frame_14 = QFrame(self.calibrationCross_2)
383         self.frame_14.setObjectName(u"frame_14")
384         sizePolicy.setHeightForWidth(self.frame_14.sizePolicy().hasHeightForWidth())
385         self.frame_14.setSizePolicy(sizePolicy)
386         self.frame_14.setMaximumSize(QSize(16777215, 20))
387         self.frame_14.setStyleSheet(u"background-color: rgb(200, 200, 200);")
388         self.frame_14.setFrameShape(QFrame.StyledPanel)
389         self.frame_14.setFrameShadow(QFrame.Raised)
390 
391         self.gridLayout_7.addWidget(self.frame_14, 1, 2, 1, 1)
392 
393         self.frame_3 = QFrame(self.calibrationCross_2)
394         self.frame_3.setObjectName(u"frame_3")
395         sizePolicy.setHeightForWidth(self.frame_3.sizePolicy().hasHeightForWidth())
396         self.frame_3.setSizePolicy(sizePolicy)
```

```
397         self.frame_3.setStyleSheet(u"background-color: rgb(200, 200, 200);")
398         self.frame_3.setFrameShape(QFrame.StyledPanel)
399         self.frame_3.setFrameShadow(QFrame.Raised)
400 
401         self.gridLayout_7.addWidget(self.frame_3, 2, 1, 1, 1)
402 
403 
404         self.gridLayout_9.addWidget(self.calibrationCross_2, 1, 1, 1, 1)
405 
406         self.horizontalSpacer_12 = QSpacerItem(486, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
407 
408         self.gridLayout_9.addItem(self.horizontalSpacer_12, 1, 2, 1, 1)
409 
410         self.verticalSpacer_11 = QSpacerItem(20, 247, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
411 
412         self.gridLayout_9.addItem(self.verticalSpacer_11, 2, 1, 1, 1)
413 
414 
415         self.gridLayout_8.addWidget(self.frame_2, 0, 0, 1, 1)
416 
417         self.promptTestWidget.addWidget(self.calibrationPromptPage)
418         self.promptPromptPage = QWidget()
419         self.promptPromptPage.setObjectName(u"promptPromptPage")
420         self.promptPromptPage.setStyleSheet(u"color: rgb(200,200,200);")
421         self.horizontalLayout_8 = QHBoxLayout(self.promptPromptPage)
422         self.horizontalLayout_8.setSpacing(0)
423         self.horizontalLayout_8.setObjectName(u"horizontalLayout_8")
424         self.horizontalLayout_8.setContentsMargins(0, 0, 0, 0)
425         self.frame_4 = QFrame(self.promptPromptPage)
426         self.frame_4.setObjectName(u"frame_4")
427         self.frame_4.setStyleSheet(u"background-color: rgb(0,0,0);")
428         self.frame_4.setFrameShape(QFrame.StyledPanel)
429         self.frame_4.setFrameShadow(QFrame.Raised)
430         self.gridLayout_10 = QGridLayout(self.frame_4)
431         self.gridLayout_10.setObjectName(u"gridLayout_10")
432         self.verticalSpacer_13 = QSpacerItem(20, 220, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
433 
434         self.gridLayout_10.addItem(self.verticalSpacer_13, 0, 1, 1, 1)
435 
436         self.horizontalSpacer_14 = QSpacerItem(384, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
437 
438         self.gridLayout_10.addItem(self.horizontalSpacer_14, 1, 0, 1, 1)
439 
440         self.rightFrame_2 = QFrame(self.frame_4)
441         self.rightFrame_2.setObjectName(u"rightFrame_2")
442         sizePolicy.setHeightForWidth(self.rightFrame_2.sizePolicy().hasHeightForWidth())
443         self.rightFrame_2.setSizePolicy(sizePolicy)
444         self.rightFrame_2.setStyleSheet(u"")
445         self.rightFrame_2.setFrameShape(QFrame.StyledPanel)
446         self.rightFrame_2.setFrameShadow(QFrame.Raised)
447         self.horizontalLayout_11 = QHBoxLayout(self.rightFrame_2)
448         self.horizontalLayout_11.setObjectName(u"horizontalLayout_11")
449         self.rightLabel_2 = QLabel(self.rightFrame_2)
450         self.rightLabel_2.setObjectName(u"rightLabel_2")
451         self.rightLabel_2.setFont(font)
452         self.rightLabel_2.setStyleSheet(u"")
453         self.rightLabel_2.setAlignment(Qt.AlignCenter)
454 
455         self.horizontalLayout_11.addWidget(self.rightLabel_2)
456 
457 
458         self.gridLayout_10.addWidget(self.rightFrame_2, 1, 1, 1, 1)
459 
460         self.horizontalSpacer_15 = QSpacerItem(384, 20, QSizePolicy.Preferred, QSizePolicy.
    Minimum)
461 
462         self.gridLayout_10.addItem(self.horizontalSpacer_15, 1, 2, 1, 1)
```

```
463
464         self.verticalSpacer_14 = QSpacerItem(20, 220, QSizePolicy.Minimum, QSizePolicy.
    Preferred)
465
466         self.gridLayout_10.addItem(self.verticalSpacer_14, 2, 1, 1, 1)
467
468
469         self.horizontalLayout_8.addWidget(self.frame_4)
470
471         self.promptTestWidget.addWidget(self.promptPromptPage)
472
473         self.horizontalLayout_7.addWidget(self.promptTestWidget)
474
475         self.demosPages.addWidget(self.promptPage)
476         self.cursorPage = QWidget()
477         self.cursorPage.setObjectName(u"cursorPage")
478         self.horizontalLayout_6 = QHBoxLayout(self.cursorPage)
479         self.horizontalLayout_6.setSpacing(0)
480         self.horizontalLayout_6.setObjectName(u"horizontalLayout_6")
481         self.horizontalLayout_6.setContentsMargins(0, 0, 0, 0)
482         self.cursorFrame = QFrame(self.cursorPage)
483         self.cursorFrame.setObjectName(u"cursorFrame")
484         sizePolicy1 = QSizePolicy(QSizePolicy.Preferred, QSizePolicy.Preferred)
485         sizePolicy1.setHorizontalStretch(0)
486         sizePolicy1.setVerticalStretch(20)
487         sizePolicy1.setHeightForWidth(self.cursorFrame.sizePolicy().hasHeightForWidth())
488         self.cursorFrame.setSizePolicy(sizePolicy1)
489         self.cursorFrame.setStyleSheet(u"")
490         self.cursorFrame.setFrameShape(QFrame.StyledPanel)
491         self.cursorFrame.setFrameShadow(QFrame.Raised)
492         self.horizontalLayout_13 = QHBoxLayout(self.cursorFrame)
493         self.horizontalLayout_13.setObjectName(u"horizontalLayout_13")
494         self.horizontalLayout_13.setContentsMargins(3, 3, 3, 3)
495         self.mouseCursor = QWidget(self.cursorFrame)
496         self.mouseCursor.setObjectName(u"mouseCursor")
497         sizePolicy2 = QSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
498         sizePolicy2.setHorizontalStretch(0)
499         sizePolicy2.setVerticalStretch(0)
500         sizePolicy2.setHeightForWidth(self.mouseCursor.sizePolicy().hasHeightForWidth())
501         self.mouseCursor.setSizePolicy(sizePolicy2)
502         self.mouseCursor.setMaximumSize(QSize(20, 20))
503         self.mouseCursor.setCursor(QCursor(Qt.ArrowCursor))
504         self.mouseCursor.setStyleSheet(u"background-color: rgb(255, 255, 255);\n"
505 "image: url(:/newPrefix/pictures/cursor.png);")
506
507         self.horizontalLayout_13.addWidget(self.mouseCursor)
508
509
510         self.horizontalLayout_6.addWidget(self.cursorFrame)
511
512         self.demosPages.addWidget(self.cursorPage)
513         self.game1Page = QWidget()
514         self.game1Page.setObjectName(u"game1Page")
515         self.horizontalLayout_9 = QHBoxLayout(self.game1Page)
516         self.horizontalLayout_9.setSpacing(0)
517         self.horizontalLayout_9.setObjectName(u"horizontalLayout_9")
518         self.horizontalLayout_9.setContentsMargins(0, 0, 0, 0)
519         self.game1Widget = QWidget(self.game1Page)
520         self.game1Widget.setObjectName(u"game1Widget")
521         sizePolicy3 = QSizePolicy(QSizePolicy.Fixed, QSizePolicy.Fixed)
522         sizePolicy3.setHorizontalStretch(10)
523         sizePolicy3.setVerticalStretch(10)
524         sizePolicy3.setHeightForWidth(self.game1Widget.sizePolicy().hasHeightForWidth())
525         self.game1Widget.setSizePolicy(sizePolicy3)
526         self.game1Widget.setMinimumSize(QSize(400, 400))
527
528         self.horizontalLayout_9.addWidget(self.game1Widget)
529
530         self.demosPages.addWidget(self.game1Page)
531         self.game2Page = QWidget()
532         self.game2Page.setObjectName(u"game2Page")
```

```
533         self.horizontalLayout_10 = QHBoxLayout(self.game2Page)
534         self.horizontalLayout_10.setSpacing(0)
535         self.horizontalLayout_10.setObjectName(u"horizontalLayout_10")
536         self.horizontalLayout_10.setContentsMargins(0, 0, 0, 0)
537         self.game2Widget = QWidget(self.game2Page)
538         self.game2Widget.setObjectName(u"game2Widget")
539         self.gridLayout_11 = QGridLayout(self.game2Widget)
540         self.gridLayout_11.setObjectName(u"gridLayout_11")
541         self.game2Label = QLabel(self.game2Widget)
542         self.game2Label.setObjectName(u"game2Label")
543         sizePolicy4 = QSizePolicy(QSizePolicy.Fixed, QSizePolicy.Fixed)
544         sizePolicy4.setHorizontalStretch(0)
545         sizePolicy4.setVerticalStretch(0)
546         sizePolicy4.setHeightForWidth(self.game2Label.sizePolicy().hasHeightForWidth())
547         self.game2Label.setSizePolicy(sizePolicy4)
548         font2 = QFont()
549         font2.setPointSize(22)
550         self.game2Label.setFont(font2)
551
552         self.gridLayout_11.addWidget(self.game2Label, 0, 0, 1, 1)
553
554
555         self.horizontalLayout_10.addWidget(self.game2Widget)
556
557         self.demosPages.addWidget(self.game2Page)
558
559         self.horizontalLayout_12.addWidget(self.demosPages)
560
561         UserWindow.setCentralWidget(self.centralwidget)
562
563         self.retranslateUi(UserWindow)
564
565         self.demosPages.setCurrentIndex(2)
566         self.promptsWidgets.setCurrentIndex(0)
567         self.promptTestWidget.setCurrentIndex(0)
568
569
570         QMetaObject.connectSlotsByName(UserWindow)
571     # setupUi
572
573     def retranslateUi(self, UserWindow):
574         UserWindow.setWindowTitle(QCoreApplication.translate("UserWindow", u"MainWindow",
    None))
575         self.rightLabel.setText(QCoreApplication.translate("UserWindow", u"Right Hand", None)
    )
576         self.leftLabel.setText(QCoreApplication.translate("UserWindow", u"Left Hand", None))
577         self.tongueLabel.setText(QCoreApplication.translate("UserWindow", u"Tongue", None))
578         self.feetLabel.setText(QCoreApplication.translate("UserWindow", u"Feet", None))
579         self.rightLabel_2.setText(QCoreApplication.translate("UserWindow", u"Right Hand",
    None))
580         self.game2Label.setText(QCoreApplication.translate("UserWindow", u"Game 2 Page", None
    ))
581     # retranslateUi
```

9.2.3. UI_SPLASHSCREEN.PY

```
1  # -*- coding: utf-8 -*-
2
3  ################################################################################
4  ## Form generated from reading UI file 'splashscreenbbLZjX.ui'
5  ##
6  ## Created by: Qt User Interface Compiler version 6.4.3
7  ##
8  ## WARNING! All changes made in this file will be lost when recompiling UI file!
9  ################################################################################
10
11 from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,
12     QMetaObject, QObject, QPoint, QRect,
13     QSize, QTime, QUrl, Qt)
14 from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
15     QFont, QFontDatabase, QGradient, QIcon,
```

```
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QFrame, QLabel, QMainWindow,
    QProgressBar, QSizePolicy, QWidget)
import image_rc

class Ui_SplashScreen(object):
    def setupUi(self, SplashScreen):
        if not SplashScreen.objectName():
            SplashScreen.setObjectName(u"SplashScreen")
        SplashScreen.resize(690, 425)
        self.centralwidget = QWidget(SplashScreen)
        self.centralwidget.setObjectName(u"centralwidget")
        self.mainFrame = QFrame(self.centralwidget)
        self.mainFrame.setObjectName(u"mainFrame")
        self.mainFrame.setGeometry(QRect(0, 54, 500, 200))
        self.mainFrame.setStyleSheet(u"background-color: qlineargradient(spread:pad, x1:0, y1
    :0.273, x2:1, y2:0.711, stop:0 rgba(6, 196, 249, 255), stop:1 rgba(0, 23, 132, 255));\n"
"border-radius: 20px;\n"
"border-bottom: 5px solid rgb(68, 161, 208);\n"
"border-right: 5px solid rgb(0, 23, 131);\n"
"border-top: 5px solid rgb(68, 161, 208);\n"
"border-left: 5px solid rgb(68, 161, 208);")
        self.mainFrame.setFrameShape(QFrame.StyledPanel)
        self.mainFrame.setFrameShadow(QFrame.Raised)
        self.logoDelft = QFrame(self.mainFrame)
        self.logoDelft.setObjectName(u"logoDelft")
        self.logoDelft.setGeometry(QRect(0, 0, 181, 151))
        self.logoDelft.setStyleSheet(u"image: url(:/newPrefix/pictures/TUDelftFlame_white.png
    );\n"
"background-color:none;\n"
"border:none;")
        self.logoDelft.setFrameShape(QFrame.StyledPanel)
        self.logoDelft.setFrameShadow(QFrame.Raised)
        self.frame = QFrame(self.mainFrame)
        self.frame.setObjectName(u"frame")
        self.frame.setGeometry(QRect(70, 10, 550, 550))
        self.frame.setStyleSheet(u"background-color:none;\n"
"border: 10px solid rgb(0, 16, 80);\n"
"border-radius:275px;")
        self.frame.setFrameShape(QFrame.StyledPanel)
        self.frame.setFrameShadow(QFrame.Raised)
        self.label = QLabel(self.frame)
        self.label.setObjectName(u"label")
        self.label.setGeometry(QRect(20, 133, 401, 71))
        font = QFont()
        font.setFamilies([u"Copperplate Gothic Bold"])
        font.setPointSize(35)
        font.setBold(True)
        self.label.setFont(font)
        self.label.setStyleSheet(u"border-top:none;\n"
"border-bottom:20px solid rgb(0, 16, 80);\n"
"border-left:none;\n"
"border-right:none;")
        self.label.setAlignment(Qt.AlignCenter)
        self.logoBorderFrame = QFrame(self.centralwidget)
        self.logoBorderFrame.setObjectName(u"logoBorderFrame")
        self.logoBorderFrame.setGeometry(QRect(440, 0, 250, 250))
        self.logoBorderFrame.setStyleSheet(u"border: 7px solid rgb(68, 161, 208);\n"
"border-radius: 20px;\n"
"background-color: qlineargradient(spread:pad, x1:0, y1:1, x2:1, y2:0, stop:0 rgba(1, 84,
    158, 237), stop:1 rgba(192, 230, 255, 255));")
        self.logoBorderFrame.setFrameShape(QFrame.StyledPanel)
        self.logoBorderFrame.setFrameShadow(QFrame.Raised)
        self.EEGFrame = QFrame(self.centralwidget)
        self.EEGFrame.setObjectName(u"EEGFrame")
        self.EEGFrame.setGeometry(QRect(440, 0, 250, 250))
        self.EEGFrame.setStyleSheet(u"image: url(:/newPrefix/pictures/EEG-removebg.png);\n"
"background-color:none;\n"
"border:none;")
        self.EEGFrame.setFrameShape(QFrame.StyledPanel)
```

```
84          self.EEGFrame.setFrameShadow(QFrame.Raised)
85          self.moreInfoFrame = QFrame(self.centralwidget)
86          self.moreInfoFrame.setObjectName(u"moreInfoFrame")
87          self.moreInfoFrame.setGeometry(QRect(1, 234, 500, 191))
88          self.moreInfoFrame.setStyleSheet(u"background-color: rgb(0, 16, 80);\n"
89  "border-radius:20px;\n"
90  "border: 5px solid rgb(68, 161, 208);")
91          self.moreInfoFrame.setFrameShape(QFrame.StyledPanel)
92          self.moreInfoFrame.setFrameShadow(QFrame.Raised)
93          self.label_2 = QLabel(self.moreInfoFrame)
94          self.label_2.setObjectName(u"label_2")
95          self.label_2.setGeometry(QRect(50, 40, 431, 51))
96          font1 = QFont()
97          font1.setFamilies([u"Rockwell Extra Bold"])
98          font1.setPointSize(20)
99          font1.setBold(True)
100         self.label_2.setFont(font1)
101         self.label_2.setStyleSheet(u"border:none;\n"
102 "color: rgb(255, 255, 255);")
103         self.label_3 = QLabel(self.moreInfoFrame)
104         self.label_3.setObjectName(u"label_3")
105         self.label_3.setGeometry(QRect(300, 90, 131, 20))
106         font2 = QFont()
107         font2.setFamilies([u"Rockwell"])
108         font2.setPointSize(16)
109         font2.setBold(False)
110         self.label_3.setFont(font2)
111         self.label_3.setStyleSheet(u"border:none;\n"
112 "color: rgb(255, 255, 255);")
113         self.label_4 = QLabel(self.moreInfoFrame)
114         self.label_4.setObjectName(u"label_4")
115         self.label_4.setGeometry(QRect(140, 160, 231, 20))
116         font3 = QFont()
117         font3.setFamilies([u"Arial"])
118         font3.setPointSize(12)
119         font3.setBold(False)
120         self.label_4.setFont(font3)
121         self.label_4.setStyleSheet(u"border:none;\n"
122 "color: rgb(154, 154, 154);")
123         self.progressBar = QProgressBar(self.moreInfoFrame)
124         self.progressBar.setObjectName(u"progressBar")
125         self.progressBar.setGeometry(QRect(70, 110, 0, 0))
126         self.progressBar.setStyleSheet(u"border:none;")
127         self.progressBar.setValue(24)
128         self.progressBar.raise_()
129         self.label_2.raise_()
130         self.label_3.raise_()
131         self.label_4.raise_()
132         #SplashScreen.setCentralWidget(self.centralwidget)
133         self.logoBorderFrame.raise_()
134         self.moreInfoFrame.raise_()
135         self.mainFrame.raise_()
136         self.EEGFrame.raise_()
137
138         self.retranslateUi(SplashScreen)
139
140         QMetaObject.connectSlotsByName(SplashScreen)
141     # setupUi
142
143     def retranslateUi(self, SplashScreen):
144         SplashScreen.setWindowTitle(QCoreApplication.translate("SplashScreen", u"MainWindow",
      None))
145         self.label.setText(QCoreApplication.translate("SplashScreen", u"WELCOME", None))
146         self.label_2.setText(QCoreApplication.translate("SplashScreen", u"Initializing BCI-
    Desk App", None))
147         self.label_3.setText(QCoreApplication.translate("SplashScreen", u"Please wait...",
    None))
148         self.label_4.setText(QCoreApplication.translate("SplashScreen", u"Designed by Group I
     (TU Delft)", None))
149     # retranslateUi
```

9.2.4. ui_ERDSWindow.py

```
# -*- coding: utf-8 -*-

################################################################################
## Form generated from reading UI file 'ERDSWindowwcdodR.ui'
##
## Created by: Qt User Interface Compiler version 6.4.3
##
## WARNING! All changes made in this file will be lost when recompiling UI file!
################################################################################

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QHBoxLayout, QMainWindow, QSizePolicy,
    QWidget)

class Ui_ERDSWindow(object):
    def setupUi(self, ERDSWindow):
        if not ERDSWindow.objectName():
            ERDSWindow.setObjectName(u"ERDSWindow")
        ERDSWindow.resize(1053, 1032)
        self.centralwidget = QWidget(ERDSWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.centralwidget.setStyleSheet(u"*{\n"
" background-color: rgb(12, 35, 64);\n"
"}\n"
"QGraphicsView{\n"
" background-color: rgb(255, 255, 255);\n"
"}")
        self.horizontalLayout_2 = QHBoxLayout(self.centralwidget)
        self.horizontalLayout_2.setObjectName(u"horizontalLayout_2")
        self.plotWidget = QWidget(self.centralwidget)
        self.plotWidget.setObjectName(u"plotWidget")

        self.horizontalLayout_2.addWidget(self.plotWidget)

        ERDSWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(ERDSWindow)

        QMetaObject.connectSlotsByName(ERDSWindow)
    # setupUi

    def retranslateUi(self, ERDSWindow):
        ERDSWindow.setWindowTitle(QCoreApplication.translate("ERDSWindow", u"MainWindow",
    None))
    # retranslateUi
```

10 Schedule

An overview of the planning over the weeks of the BAP.

Table 10.1: Distribution of the work over the weeks

Week	Task
Week 1	Complete literature list.
Week 2	Complete all necessary research, draft program of requirements, devise an implementation plan per subgroup.
Week 3	Work on subsystems, finalize program of requirements.
Week 4	Work on subsystems, prototypes of individual subsystems, work on experiment setup.
Week 5	Work on subsystems, complete individual subsystems, start integrating, submit intermediate report.
Week 6	Work on subsystems, complete individual subsystems, continue integrating.
Week 7	Work on subsystems, start finalizing integrating, continue integrating.
Week 8	Work on entire system, continue integrating, submit thesis.
Week 9	Work on entire system, finalize integrating.
Week 10	Work on entire system, finalize integrating, defend thesis and business plan.

11 GUI

11.1. Splashscreen

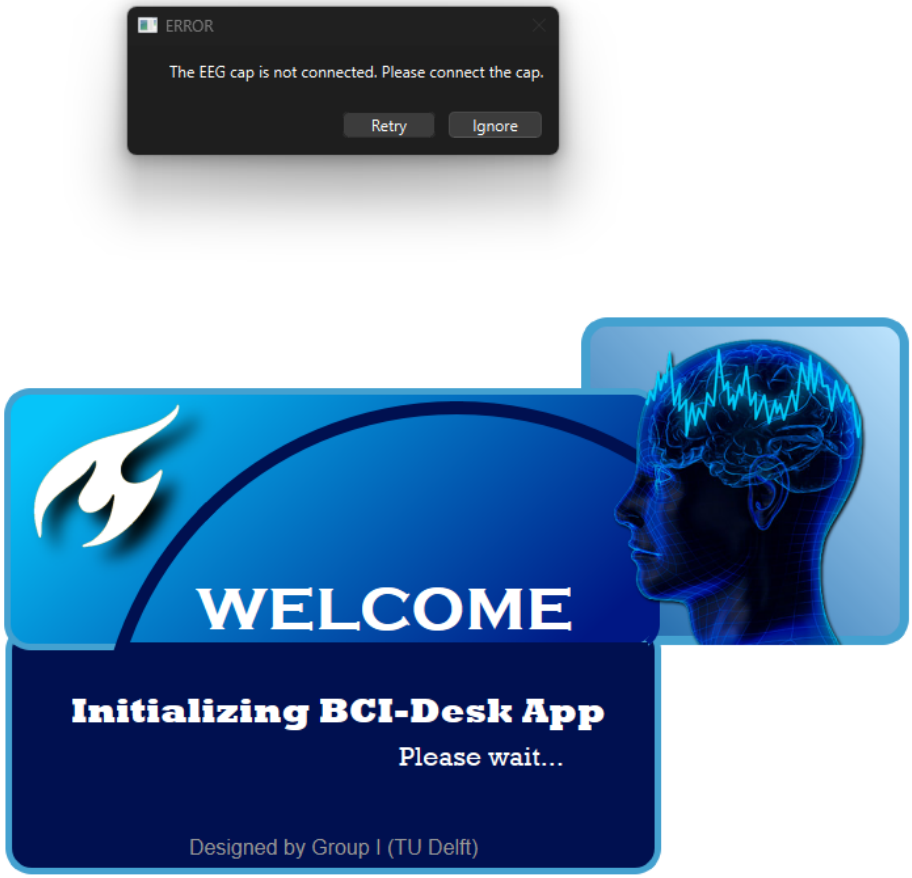


Figure 11.1: Splashscreen before opening Main Window with error saying it's not connected to the EEG cap.

11.2. Main Window

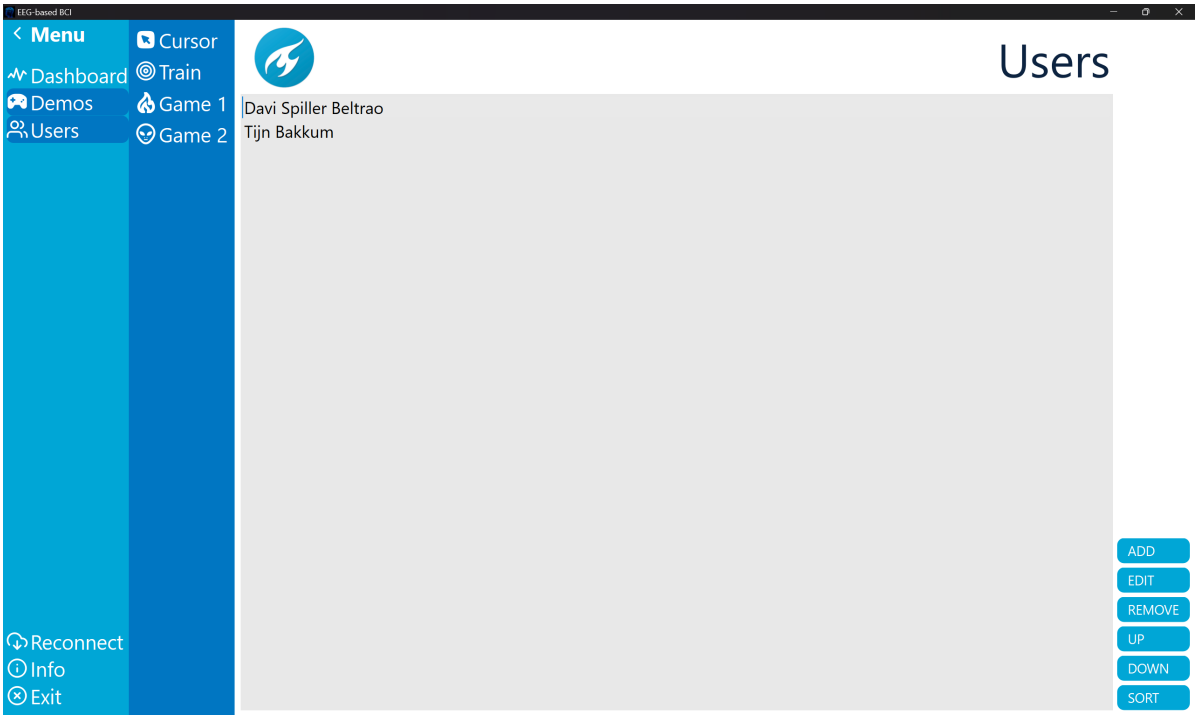


Figure 11.2: User page

Figure 11.3: Information widget when clicking on info button.

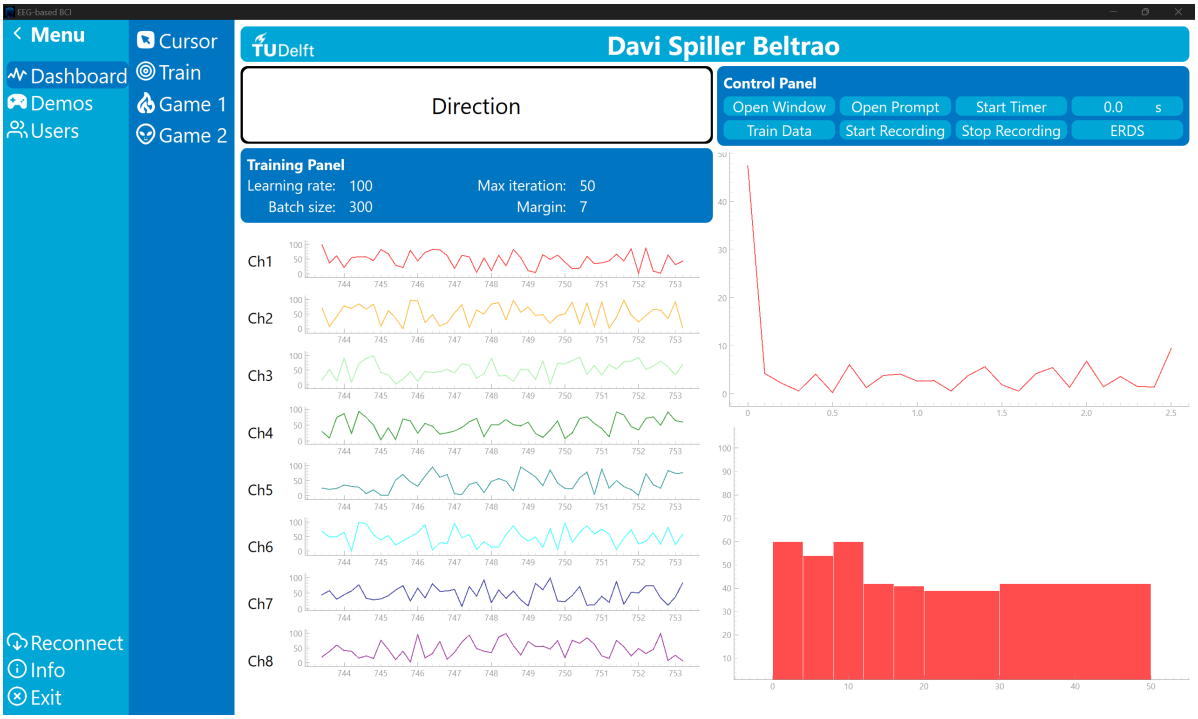


Figure 11.4: Dashboard page with random data.

Figure 11.5: Dashboard page connected to cap.

11.3. User Window

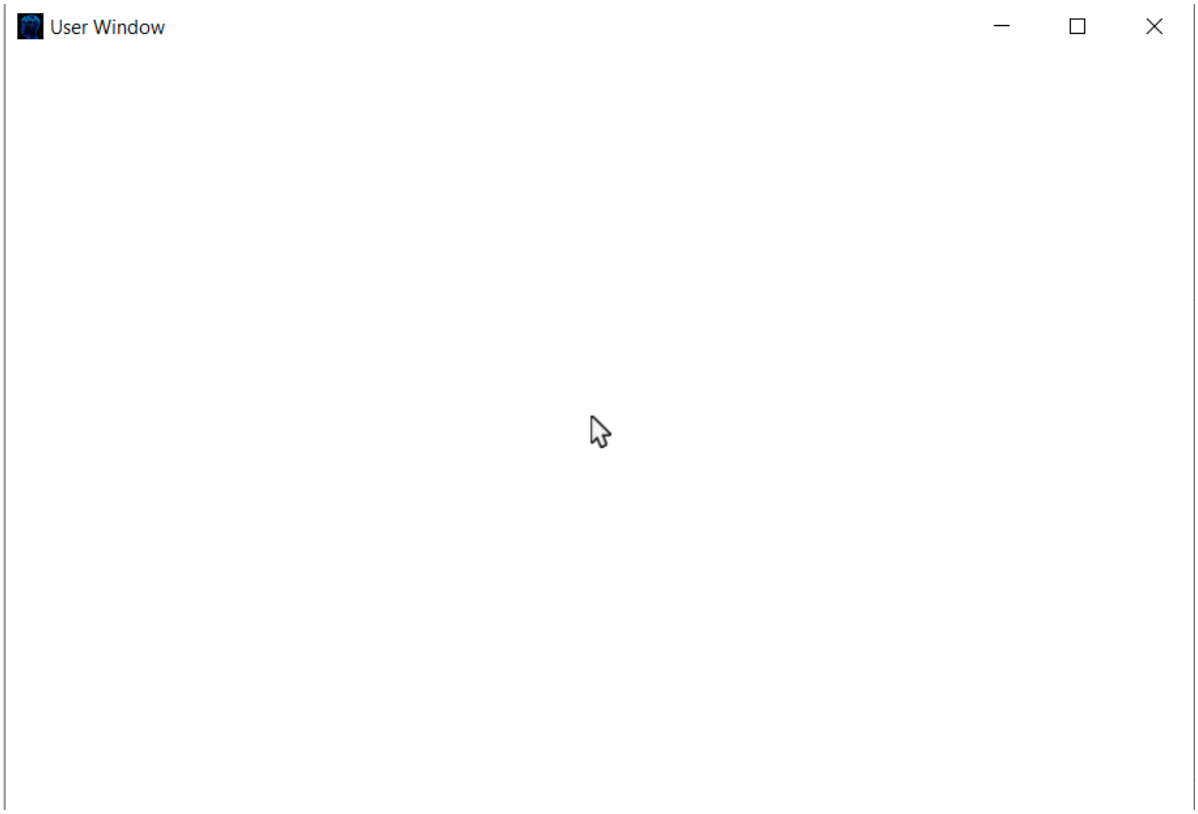


Figure 11.6: Cursor demo

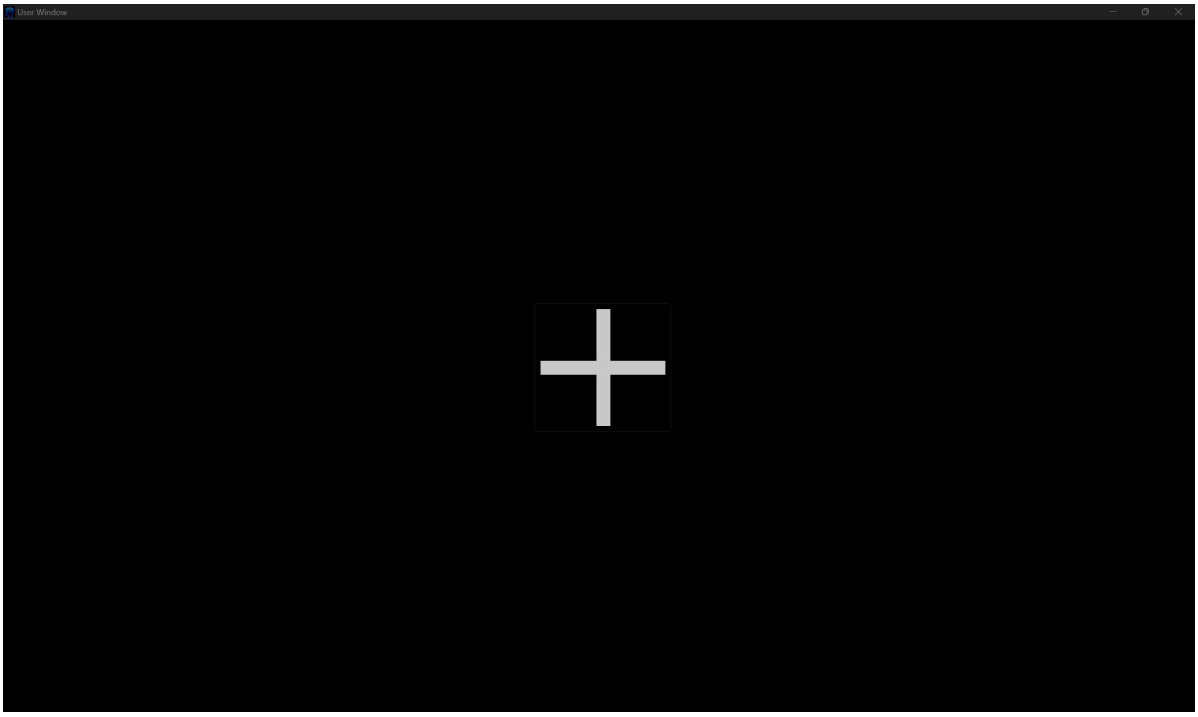


Figure 11.7: Calibration cross

Figure 11.8: Feet prompt

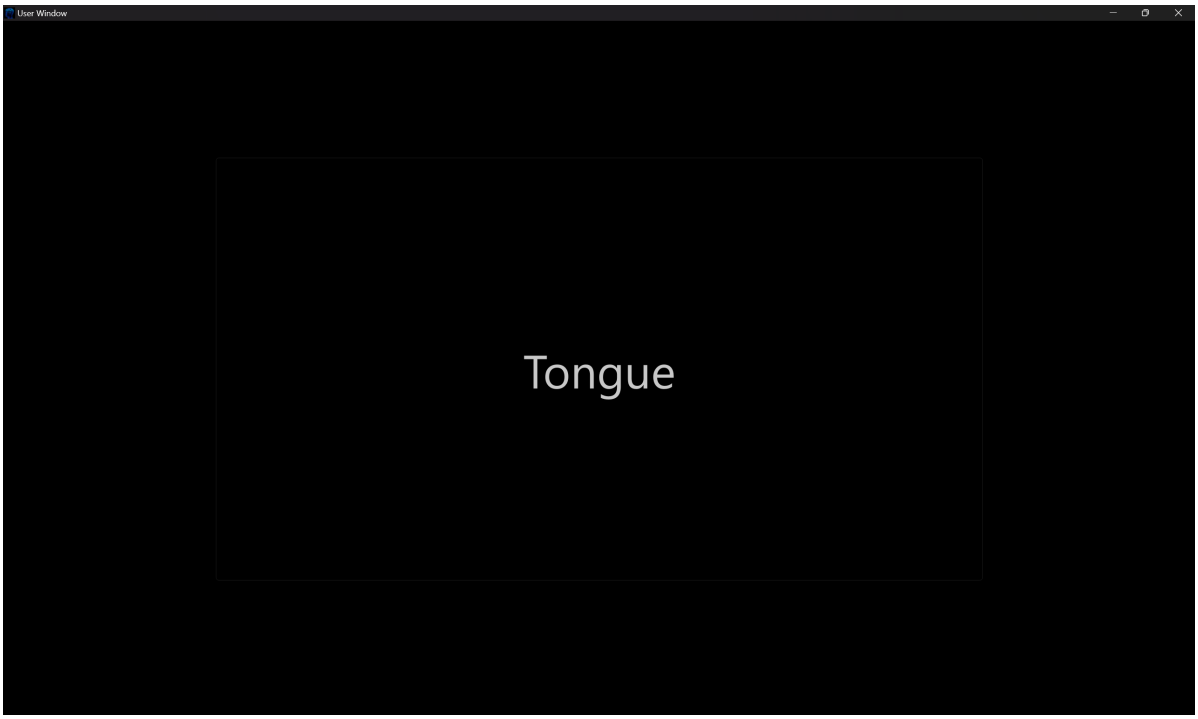


Figure 11.9: Tongue prompt

Figure 11.10: Left hand prompt

Figure 11.11: Right hand prompt

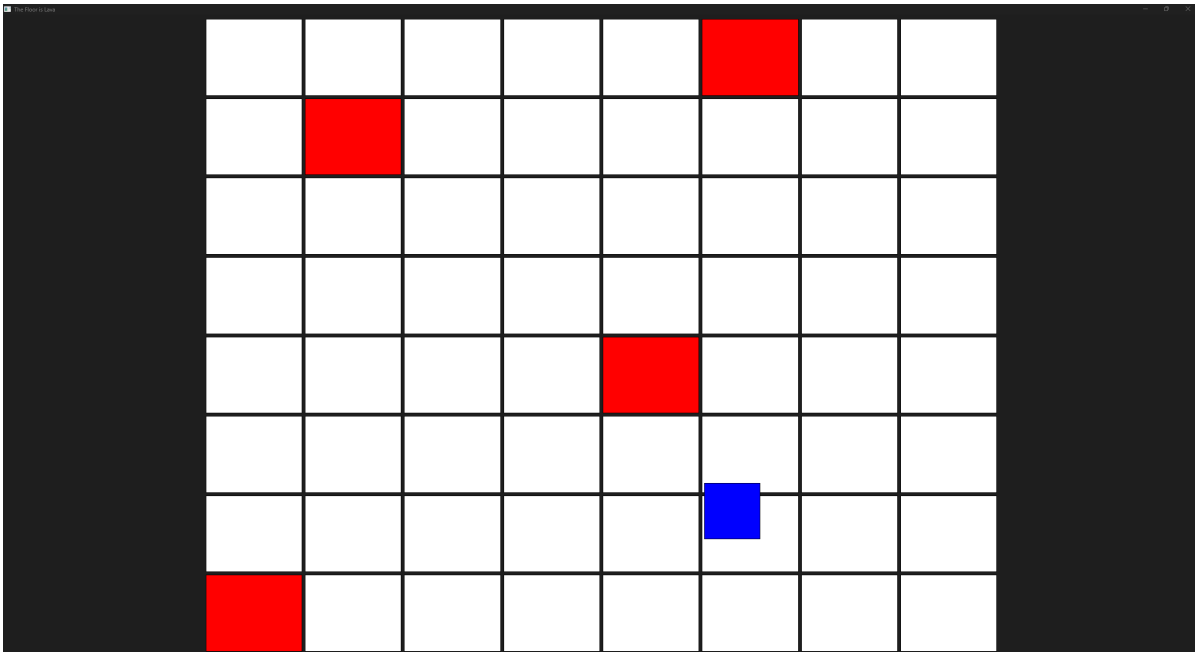


Figure 11.12: Floor is Lava game

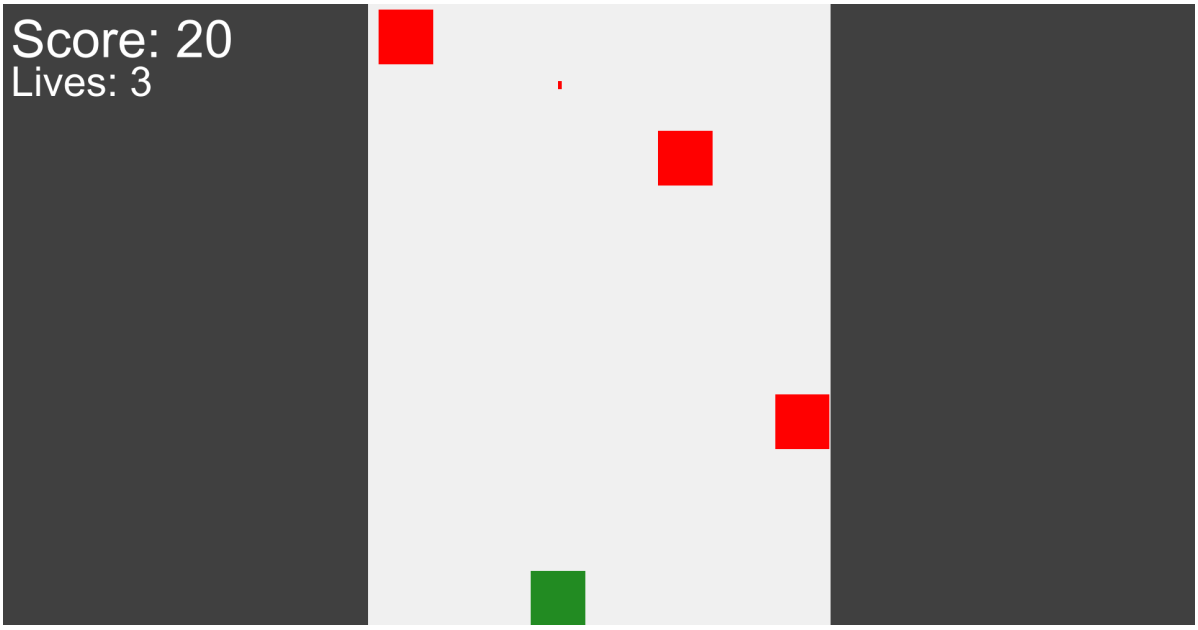


Figure 11.13: Meteor Shooter game

11.4. One screen setup

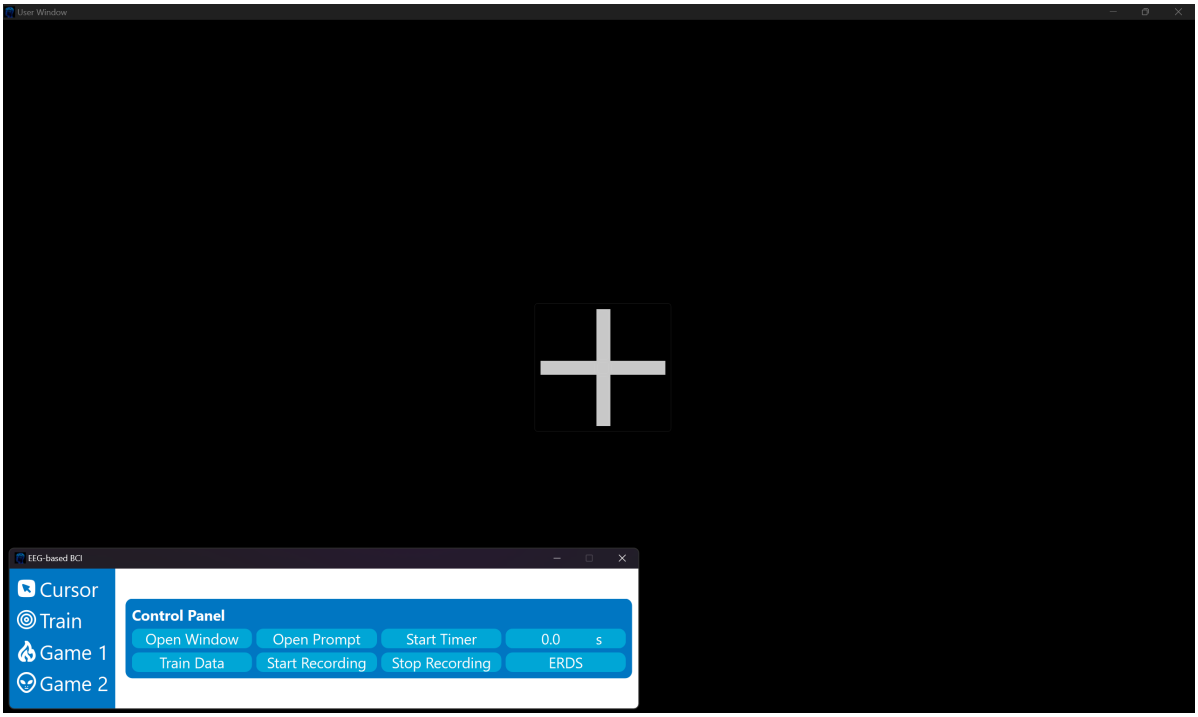


Figure 11.14: Recording while on User Window with shrunk Main Window with the control panel and demos submenu open.