UCLouvain

École polytechnique de Louvain

Building a GUI to streamline paradigm creation and presentation in psychology

Author: **Corentin Warzée**
Supervisors: **Benoît Macq, Laurence Dricot**
Readers: **Nicolas Delinte, Sébastien Jodogne, Ron Kupers**
Academic year 2024–2025
Master [120] in Computer Science

Abstract

This thesis details the creation of an intuitive application for the design and execution of psychological paradigms in clinical and research settings. The program addresses the deficiencies of current technologies, such as complexity, cost, and limited adaptability, by offering a tool that is user-friendly and efficient.

The application supports paradigms such as Stroop and Cyberball tests, with features such as speech recording, precise key-press synchronization and dynamic stimulus presentation. It produces output files specially designed to be compatible with BrainVoyager, facilitating data processing. Modular architecture enables adaptation to a variety of research and processing requirements.

Developed in partnership with the Institute of Neuroscience at UCLouvain, the tool addresses practical needs, rendering it an invaluable resource for physicians and researchers. It facilitates the development of complex paradigms without sacrificing usability and promotes new avenues for investigating cognitive and emotional processes. This project bridges the gap between clinical needs and scientific innovation.

Acknowledgements

Throughout the process of writing this thesis, I received invaluable support and assistance from numerous individuals, without whom this work would not have been possible. I would like to express my heartfelt gratitude to all of them.

Firstly, I would like to sincerely thank my promoters, Benoît Macq and Laurence Dricot, for their unwavering guidance and encouragement throughout this journey. Their expertise and insights were essential to the completion of this thesis. I am particularly grateful to Laurence for her private lessons, which introduced me to the medical concepts that were fundamental to my dissertation and were entirely new to me.

I would also like to thank Quentin Dessain, Nicolas Delinte and Colin Vanden Bulcke for their advice. Their views and suggestions helped refine my work.

A special note of thanks goes to Ron Kupers, who was a constant source of support throughout this entire process. His availability, dedication, and willingness to answer my many questions were truly invaluable. He not only provided guidance but also helped me overcome numerous challenges, making this work possible.

Lastly, I want to thank my family and friends, whose unwavering support and encouragement have been a cornerstone of my academic journey. Their belief in me and their constant presence have been incredibly meaningful throughout this journey.

The Deepl tool was used to help me translate sentences. ChatGPT was used for text structuring and proofreading.

Contents

1 Introduction

The examination of mental activity and cognitive processes is largely influenced by the paradigms used in psychology. According to the Larousse dictionary, a psychological paradigm is "a methodological procedure that serves as a reference model" [9]. More concretely, it is a set of theories, methodologies and concepts used collectively by a scientific community to examine mental behavior. These paradigms serve as theoretical and methodological frameworks for analyzing complex issues in clinical psychology and neuroscience. Thomas Kuhn argues that paradigms standardize approaches and guide researchers in identifying problems and formulating answers [5].

These paradigms are frequently modified, as in the Stroop or Cyberball models, to meet unique needs. This is particularly true in clinical environments, as patients present a wide range of medical characteristics. On the other hand, the tools currently available to build and present these paradigms present substantial constraints that hinder their use and effectiveness. Even if functional, currently available software solutions are costly, have excessively complicated user interfaces and regularly encounter technical difficulties.

In the case of Clinique of Saint-Luc, for example, users have reported problems with the fluidity of visual stimuli, errors in protocols and an excessive number of triggers that make data analysis difficult. As a result, some patients have to be redirected, causing bottlenecks in ongoing clinical operations. These tools often lack essential functionalities, such as the ability to record verbal responses or to demonstrate certain paradigms like the Cyberball. Important functionalities are generally absent from these programs, which is another drawback. The fact that the diffuclty of use of such software implies dependence on a limited group of specialists to correct or configure the paradigms makes them even more difficult to manage. As a result, this ultimately leads to a restriction of the user's independence and autonomy.

This thesis proposes the development of a new application for clinical and research environments with the aim of simplifying the construction and presentation of psychological paradigms in response to the problems that have been presented in this statement. It is designed to address key issues such as the smooth presentation of visual and auditory stimuli, the customization of implemented paradigms to meet specific patient needs, the integration of functions such as the recording of vocal responses and the creation of personalized paradigms. This application was developed in collaboration with Clinique of Saint-Luc and the Institute of Neuroscience, and is designed to answer these key questions.

Priority has been given to a simple user interface to ensure that the program is easily accessible and user-friendly.

Although the requirements of Clinique of Saint-Luc were the main driving force behind the application's development, its architecture was designed with adaptability in mind, enabling it to be used in a variety of other clinical contexts. For example, the auditory paradigm used in this study was modified for use at the Antwerp UZA, demonstrating the adaptability of this phenomenon in a variety of contexts. In addition, the software has also been used effectively with patients in Copenhagen, demonstrating its broad applicability and potential as a universal tool for researchers and clinicians alike.

Starting with a survey of the idea of paradigm in psychology, this thesis will then explain the theoretical and practical significance of these paradigms. It will then examine the particular paradigms that have been incorporated into the application, such as Stroop and Cyberball, as well as auditory and visual models. A discussion will also be held on the technical difficulties encountered throughout the development of the application, and the solutions applied to improve its reliability and robustness.

This application represents a major step forward in the integration of high-performance, accessible tools. It does so by merging the requirements of practitioners and patients with advances in technology. It not only meets pragmatic challenges, but also expands the range of possibilities offered by psychological paradigms. This innovation not only has obvious implications for clinical practice, but also enables the acquisition of more reliable and diverse experimental data, opening up new avenues of research into fundamental aspects of the brain. Given the resulting convergence of technology and the human sciences, it's important to emphasize the importance of solutions that enhance both clinical practice and research.

2 Paradigm in psychology

In psychology, a "paradigm" is a set of accepted ideas, theories, ideals, and methods that a scientific community uses to direct its investigations and procedures. This framework determines the instruments and criteria used to evaluate and validate results in addition to defining the field's scope of investigation. Paradigms give scientists a shared conceptual and methodological basis, which gives their work coherence.

2.1 Historical background and evolution

The term "paradigm" was first presented by Thomas Kuhn in his seminal work, "The Structure of Scientific Revolutions," which was published in 1962 [6]. The analysis presented by Kuhn presented a challenge to the conventional understanding of scientific advancement as a sequential and cumulative process. In its place, he put forward the idea that scientific progress is achieved through a sequence of paradigm changes. As a result of the fact that each paradigm rules a certain period of "normal science," researchers are able to work within the confines of the recognized framework, therefore resolving mysteries and enhancing the predictions of the paradigm.

However, when the number of anomalies, which are events that the present paradigm either does not adequately explain or does not adequately address, increases, faith in this paradigm begins to weaken. The current crisis lays the way for a scientific revolution, which will result in the emergence of a new paradigm and the redefinition of the fundamental assumptions that underpin the discipline. As an illustration, the transition from behaviorism to cognitive psychology is a noteworthy example in the field of psychology. Behaviorism, which was prevalent in the early 20th century, was a school of thought that disregarded the study of mental processes as speculative and instead concentrated solely on observed behavior. The limits of behaviorism were brought to light by developments in experimental methods and theoretical models by the middle of the 20th century. This led to the development of cognitive psychology, which places emphasis on the mental processes that occur within the individual, such as perception, memory, and thinking.[10]

2.2 The Role of Paradigms in Psychological Research

In the field of psychology, paradigms serve multiple essential functions:

- **Theoretical Framework:** A paradigm serves as the lens through which researchers comprehend and explore psychological phenomena. For instance, psychodynamic frameworks analyze behavior by considering unconscious processes and formative developmental experiences, while biological paradigms focus on neurochemical and genetic factors [14]
- **Methodological Guidance:** Paradigms delineate the permissible methods and instruments for executing research. For example, the behaviorist framework was predominantly based on controlled experiments and quantifiable results, whereas modern frameworks, including cognitive neuroscience, utilize sophisticated technologies such as functional magnetic resonance imaging (fMRI) and electroencephalography (EEG) to examine the connections between brain activity and behavior.[13]
- **Criteria for Evaluation:** Paradigms set the benchmarks for evaluating the validity, reliability, and significance of research outcomes. The specified criteria guarantee that research undertaken within this framework is consistent with its foundational assumptions and methodologies.

2.3 Paradigms in Psychological research

Kuhn's framework significantly influences the comprehension of the development of psychological science. Various paradigms within psychology frequently exist simultaneously, illustrating the discipline's diversity and interdisciplinary characteristics. Clinical psychology, for instance, synthesizes paradigms from psychodynamic, cognitive-behavioral, and biological traditions, among others, to effectively address mental health issues. Every paradigm provides unique perspectives, techniques, and strategies, highlighting the significance of diverse approaches in comprehending intricate human behavior.

Furthermore, the advent of new paradigms frequently mirrors wider societal and technological transformations. The incorporation of neuroscience within the field of psychology has resulted in the emergence of cognitive neuroscience, effectively connecting established psychological theories with biological information acquired through brain imaging methodologies [13]. In a similar vein, the growing acknowledgment of cultural and contextual influences on behavior has led to the development of frameworks such as cultural psychology and ecological systems theory, which highlight the interaction between individuals and their surroundings.

2.4 Paradigm Shifts in Psychology Examples

A number of paradigm changes have influenced psychology's development. The shift from structuralism to functionalism in the late 19th century is one such example. Through introspection, structuralism, supported by Wilhelm Wundt and Edward Titchener, sought to examine the fundamental components of consciousness. However, this approach has faced criticism for its subjective and reductionist methodologies. Functionalism, spearheaded by William James, arose as a reaction, emphasizing the function of mental processes and their importance in responding to environmental demands. This transition established the foundation for applied psychology and experimental inquiry [10].

A notable paradigm change occurred in the mid-20th century with the emergence of humanistic psychology. In opposition to the determinism inherent in psychodynamic and behaviorist paradigms, humanistic psychologists such as Carl Rogers and Abraham Maslow highlighted the concepts of free choice, personal development, and self-actualization [12]. This paradigm introduced a more comprehensive and positive point of view to the study and treatment of psychological disorders, which affected practices in education, counseling, and organizational development.

2.5 Modern Interactions within the Paradigm

Today, psychology is distinguished by a variety of paradigms that may interact and cross over. Integrative techniques, including biopsychosocial models, for example, mix aspects from biological, psychological, and social perspectives to offer complete explanations for behavior. This plurality captures the complexity of human behavior and the need for many points of view to address psychological concerns.

Still, cohabitation of several paradigms also has difficulties. Variations in basic assumptions, techniques, and vocabulary could hinder multidisciplinary cooperation and integration. Dealing with these difficulties calls for continuous communication and attempts to heal conceptual and methodological gaps.

2.6 In summary

Advancement of psychology science depends on paradigms, which also shape how researchers view, probe, and interpret human behavior. From Kuhn's insights into the dynamics of scientific revolutions to the modern interaction of several paradigms, the idea remains fundamental to grasp the development and practice of psychology. Examining the historical and continuous changes in paradigms helps psychologists to see the depth and flexibility of their profession in addressing the complexity of human experience. [7]

3 Paradigm implemented

Having examined the concept of paradigms in psychology and their significance, I will now provide a detailed overview of the paradigms utilized in the application, their distinct goals, and the parameters necessary for their implementation. To ensure clarity, the section is partitioned into three subsections:

- The first subsection introduces each paradigm, describing its functionality and clinical relevance.
- The second subsection explains the input parameters available, detailing their roles and how they influence the paradigms.
- The third subsection maps each paradigm to its corresponding input parameters, specifying what is required to configure and execute them.

The paradigms discussed here have been carefully selected to meet the clinical demands identified throughout the thesis. Additionally, if you would like to see a visual representation of the interface for entering parameters, refer to Figure 19 in Section 6, which provides further insight into this process.

3.1 Description of implemented paradigms

3.1.1 Artificial intelligence image

The Artificial Intelligence Images paradigm seeks to investigate the processes through which the brain analyzes and categorizes visual data. Participants are presented with images from 36 distinct object categories during this activity. The images are displayed in a pseudo-random order (can be activated or deactivated), generating an appearance of randomness while ensuring an equitable distribution of item categories throughout the session.

The display of the images use a "flashing" technique. Each image is displayed multiple times within a single stimulus period, remaining on the screen for 0.4 seconds before vanishing for 0.1 seconds. This cycle persists until the complete duration of the stimulus, generally lasting several seconds, is attained.

Participants are directed to passively view the visuals displayed on the screen, with no particular action necessitated. This configuration emphasizes the brain's inherent visual processing capabilities, free from external job interference. The task's simplicity guarantees that the obtained data accurately represents the brain activity linked to visual identification and categorization.

The objective of the paradigm is to employ machine learning methods to evaluate recorded brain activity and identify the exact object category observed by the participant. Researchers seek to decode the brain's differentiation of various object kinds by connecting neural response patterns with given visuals. This method emphasizes the capability of integrating neuroscience with artificial intelligence to derive significant insights from intricate neural data.

This approach is particularly advantageous for understanding the essential mechanisms of object identification and the brain's arrangement of visual information. The results have substantial implications, establishing a foundation for exploring cerebral networks associated with perception and facilitating the development of advanced tools for brain research and analysis.

Figure 1: Representation of a stimulus that can be presented in this paradigm

3.1.2 Artificial intelligence audition

In this paradigm, participants are instructed to execute a task without visual input to investigate auditory processing and cognitive imagery. Rather than viewing visuals, participants receive aural descriptions of objects (for instance, 'a blue backpack' or 'a bunch of strawberries') from one of 36 unique object categories. After receiving the description, participants are asked to mentally visualise the object described. The aim of this activity is to elucidate how the brain interprets and represents objects using linguistic cues, by integrating auditory data with mental images.

The paradigm operates as a cycle. The process begins with a beep, succeeded by a short interval before the participant perceives the stimulus. A brief delay ensues, after which the participant is directed to envision a fixation cross. This action functions as a cognitive reset, eliminating residual pictures from the prior trial and setting a uniform baseline for the subsequent one. Following a concluding interval of inactivity, the loop begins again.

The major aim of this paradigm is to employ machine learning techniques to interpret brain activity and anticipate the nature of the participant's imagination. Researchers aim to elucidate the mechanism by which the brain transforms verbal descriptions into visual representations by analyzing neural responses to auditory stimuli and the associated mental imagery. This paradigm illustrates the ability of artificial intelligence to interpret complex brain signals and provides essential insights into the intricate link between auditory information processing, mental imagery, and item classification.

For this paradigm, given that it has been designed for blind patients and that they are asked to imagine what is being heard throughout the experiment, there is only a gray background with nothing on it.

3.1.3 Static images

In this paradigm, static images of objects, bodies or faces are displayed on screen. Each image is presented either in its original, unchanged form, or in a scrambled form. In general, each image is displayed for one second, but this duration can be modified to suit the needs of the study or the participant, thanks to the customizable options available in the application. The assortment of bodies and faces covers a wide range of emotional expressions, including fear and joy, facilitating the study of emotional processing.

Participants are initially invited to examine the stimuli attentively, without taking any specific action in response to the images. Throughout the activity, an image with a blue background appears at irregular intervals to keep participants interested and engaged. Participants are encouraged to press a button at the highest possible speed whenever this happens. The aim of this sporadic engagement is to ensure that participants are attentive throughout the activity.

Faces, bodies and objects are the three categories of stimuli that this paradigm seeks to identify and delineate. The main aim of this paradigm is to identify and delineate brain areas that exhibit unique responses to these stimuli. To understand the brain's processing of these categories and the potential arrangement of these categories in various domains, this distinction needs to be well understood. This paradigm enables us to better understand the processes of perception and emotional recognition by manipulating stimuli and examining responses.

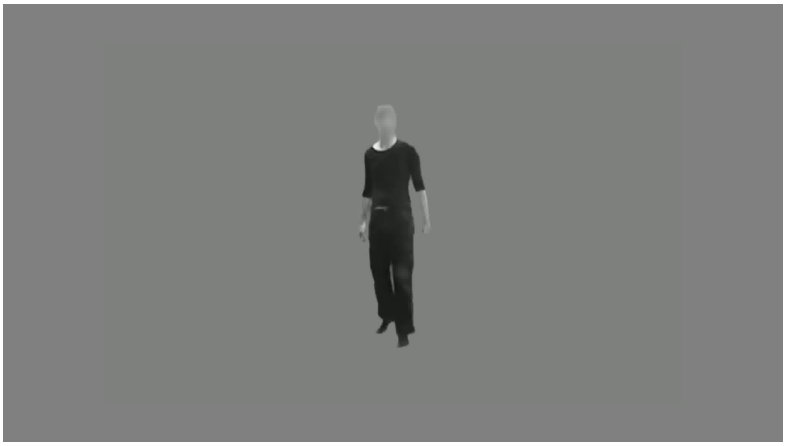

Figure 2: Representation of a stimulus showing a body

3.1.4 Scrolling videos

This paradigm uses the same concept as that of the static image, except that here it presents stimuli in motion. As with static images, images of objects, faces and bodies will be shown each time for 1 second. The goal is to explore how incorporating movement affects brain activity in regions responsible for processing these categories.

The addition of motion makes the task more complex, activating the brain's systems that help us perceive and understand dynamic information. This lets researchers look at how people respond to both static and dynamic stimuli, showing how the brain can adapt to motion. Looking at these differences helps us understand how our visual and perceptual systems work together to recognize faces, bodies, and objects in motion.

Figure 3: Representation of a stimulus

3.1.5 Positive negative adjectives

Before the paradigm begins, as with the others, instructions are shown on the screen. However, unlike the others, this paradigm includes a training phase to allow the patient to become familiar with the task. Once the training phase is complete, the actual paradigm begins by displaying an instruction to the patient.

There are three categories of instructions. The first is "ME", where the patient must determine how well the presented word characterizes them. The patient will hold two joysticks, with each joystick button corresponding to a number between 1 and 4, where 1 indicates that the word does not represent them at all, and 4 indicates that the word fits perfectly. Each stimulus is presented within a block, which consists of multiple stimuli displayed sequentially. If the patient clicks a button before the stimulus duration elapses, the stimulus disappears, and the screen remains blank until the end of the scheduled duration, after which the next stimulus appears.

The second category is "MY FRIEND", which functions in the same way as the "ME" category, except that the responses apply to the patient's best friend or closest person.

Finally, the third category, "SYLLABLE", involves a completely different task: the patient must state how many syllables the presented word contains.

Figure 4: Representation of a Stimulus in The positive/negative adjectives paradigm

3.1.6 Audition

This paradigm focuses on the patient's hearing and vocal cords. During the experiment, the patient will be shown four different symbols, each indicating whether they will hear their own voice, hear nothing, say "A" for a specified duration, or remain silent. These symbols can be categorized into two groups: the first is auditory (a symbol indicating that the patient will hear a sound and a symbol indicating that they will hear nothing), and the second is vocal (a symbol indicating that the patient must produce a sound and a symbol indicating that they must remain silent).

The patient will see one symbol from each group displayed together on the same frame, along with a countdown indicating the time remaining before performing the actions associated with these symbols. After the countdown is completed, a loading bar will appear and gradually fill. This bar represents the duration during which the patient must perform the actions indicated by the previously displayed symbols.

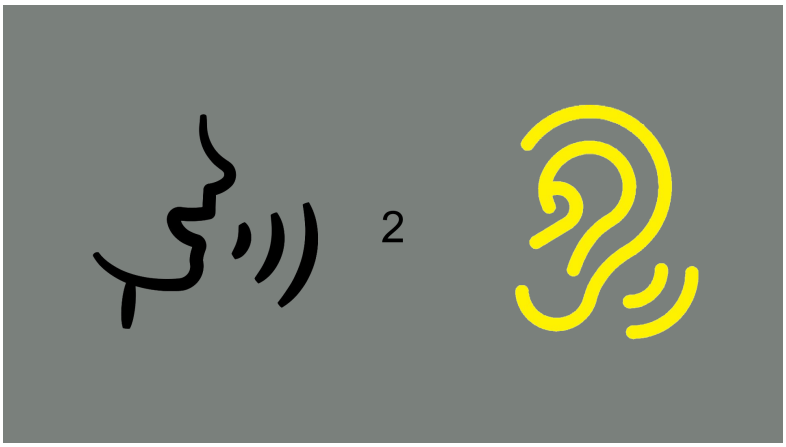

Figure 5: Representation of action to be taken when countdown is complete

3.1.7 Cyberball

The aim of this paradigm is to measure social exclusion. To simulate this, when the patient is ready to start his session, he will find himself in front of a waiting screen like in a game, with dotted lines moving to show that there is a progression in the loading and that it is not frozen or buggy. This screen serves to make him think that he is not alone, and that when he is ready, the others are not necessarily ready yet. Once this waiting phase is over, he can finally start playing with the

"other two players". The game consists of 3 players passing the ball to each other. Each time the patient's character receives the ball, he has the choice of passing it to the player on his left or to the player on his right. As time goes by, the patient will start to receive fewer passes from the other two players, until he reaches a stage where he will not receive the ball at all.

For this paradigm, there are 3 phases, the first phase, during which the game proceeds normally, i.e. bots have the same probability of sending the ball to another bot or to the patient, the transition phase, during which bots become progressively less likely to send the ball to the patient, and finally the exclusion phase, in which the probability of a player sending the ball to the patient becomes equal to 0. To calculate the time it takes the computer to send the ball back, the Python package random is used, which generates a uniform distribution between the minimum and maximum reaction time the computer has to send the ball back.

Of course, when the patient is in the scanner room, the screen on which the neurologist or technologist is working is mirrored, which poses a problem because the patient would be able to see the parameters being entered. However, the Clinique of Saint-Luc has a small box that allows the technologist to simply press a button to hide the content of the computer screen from their view.

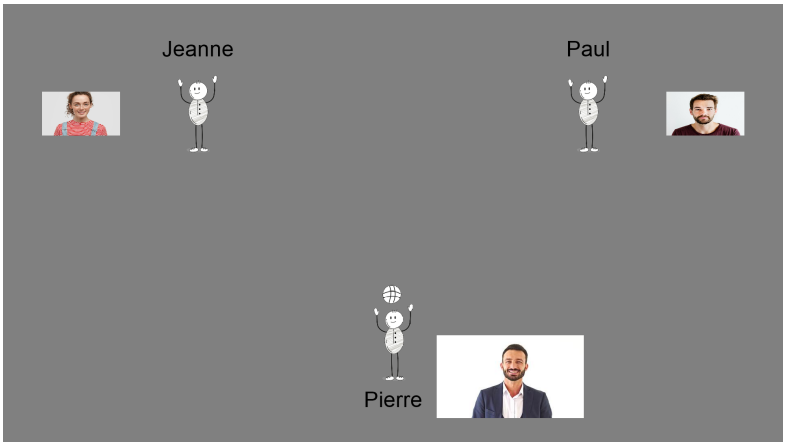

Figure 6: Patient has the ball in

3.1.8 Emo Faces

This paradigm makes use of the well-known dataset that was developed by Ekman and Friesen in 1976 and is referred to as Pictures of Facial Affect. The photos contained inside this paradigm illustrate various expressions of emotion. The photographs show a variety of men and women showing a wide range of feelings, including happiness, rage, sadness, and indifference. Participants need to look closely at these images.

Sometimes, an image is shown with a different background color. When this happens, participants are told to press a button as fast as they can. This extra task helps keep focus during the experiment and gives a way to measure how quickly someone reacts to surprising stimuli.

The main aim of this paradigm is to investigate how various emotional expressions affect neural and behavioral responses. By systematically showing these expressions and adding an interactive part, this approach helps to understand how the brain processes emotional facial cues and reacts to changes in visual patterns.

Figure 7: Stimulus presented while the execution of the paradigm

3.1.9 Emo Voices

Participants are given the opportunity to listen to brief audio sequences of a fictional language, which are delivered by actors who are trained in the field. There are a variety of feelings that are communicated through each sequence, including happiness, sadness, rage, and disgust. According to Banziger, Mortillaro, and Scherer [2], the audio stimuli were obtained from the GEMEP (Geneva Multimodal Emotion Portrayals), which is a standardized database developed for the purpose of conducting research on the perception of other people's emotions.

The participants are given the instruction to hit a button anytime they hear a voice expressing disgust while they are participating in the task. Through the use of this interactive component, researchers are able to evaluate the participants' capacity to identify particular feelings predicated on their vocal prosody. Responses to pressing the button are recorded for the purpose of analyzing reaction times and correctness in detecting the emotion that is being targeted.

The utilization of a language that does not exist guarantees that the participants will concentrate entirely on the emotional tone of the speech rather than the semantic substance of the verbal exchange. Utilizing this paradigm, one can gain useful insights into the manner in which the brain processes and recognizes emotional cues that are sent through vocal expressions.

Throughout the paradigm, when a stimulus begins to play, a representation of an ear in white is also shown.

Figure 8: Presentation of a stimulus

3.1.10 Localizer

Images of houses, bodies, and faces are displayed on the screen in the Localizer paradigm. For this assignment, as opposed to the paradigms that were discussed earlier, stimuli are organized into blocks, and each block is comprised of nine images that belong to the same category (for example, all houses, all faces, or all bodies respectively). A pause of one second is used to separate each image that is contained within a block. To serve as a point of reference for comparison, a fixation cross is displayed throughout the twelve-second gap that occurs between each succeeding block. Of course, the number of stimuli per block is adjustable, as are all the timings mentioned above.

Differentiating the activity of the brain in areas that are linked with the processing of particular visual categories is the major objective of this paradigm. The fusiform face area (FFA) is engaged in facial identification, while the extrastriate body area (EBA) is linked to the perception of body-related stimuli. For example, the parahippocampal place region (PPA) is associated with processing places, and FFA is involved in facial recognition. Using this paradigm, detailed mapping of these specialized brain regions is made possible by isolating each category into its own respective block.

Figure 9: Representation of a stimulus in a block

3.1.11 Scrolling Words

For this paradigm, the user will have to enter several values: the duration of the stimuli in seconds, the duration of the fixation cross, the zoom on the words to be presented, whether these words will be presented in a random order relative to the given order, and finally either a text file containing the list of words to be displayed or a field in which they can enter all the words to be displayed, ensuring to place a comma between each word.

The paradigm consists of a black background on which symbols that typically evoke nothing in the patient (such as "#$µ*¨ç") will first scroll, ensuring the patient is not thinking of anything. After this phase, a fixation cross will be displayed for the duration specified when the paradigm was launched. Then, a new phase will begin, where words (stimuli) will scroll one after the other. Afterward, the fixation cross will reappear, followed by a final phase where symbols evoking nothing will again scroll on the screen.

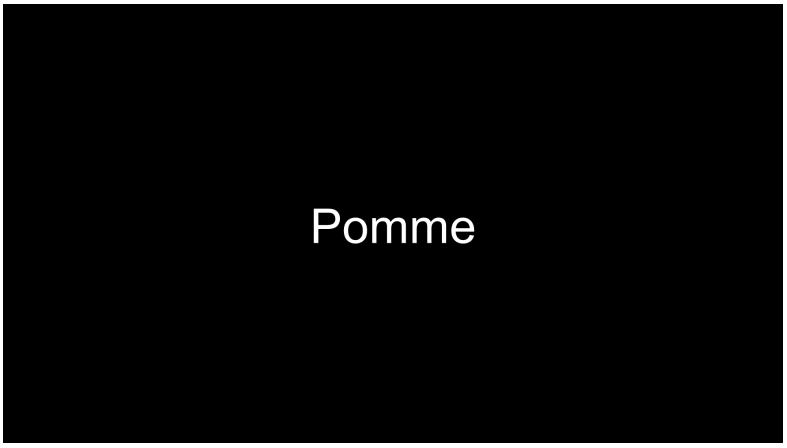

Figure 10: Stimulus representation in the Scrolling words paradigm

3.1.12 Repetition Priming

Images from the same categories that were utilized in the Static Images paradigm are given in this paradigm with the same structure. The stimuli are displayed in pairs of two, with a one-second gap between each image. Stimuli pairs are either the association of two stimuli of the same category (e.g. two stimuli representing faces) or of stimuli of different categories (e.g. a body and an object).

It is instructed to the participants that they should hit a response button whenever the images in a pair belong to separate categories (for example, a body followed by a face), but they should not press a response button when both images belong to the same category (for example, two objects or two faces). The recordings of the participants' responses provide the researchers with the opportunity to assess the participants' capacity to differentiate between categories and identify variations in the types of stimuli that were provided.

The investigation of the neurological correlates of repeated priming is the primary objective of this paradigm. The phenomenon that describes how the brain's response to a stimulus is diminished when the same or a comparable stimulus is delivered again in rapid succession is referred to as the response reduction phenomenon. This paradigm helps to uncover how the brain interprets repeated exposure to recognized categories and how it differentiates between new and recurring inputs by comparing the activity of the brain in response to repeated stimuli to that of non-repeated stimuli.

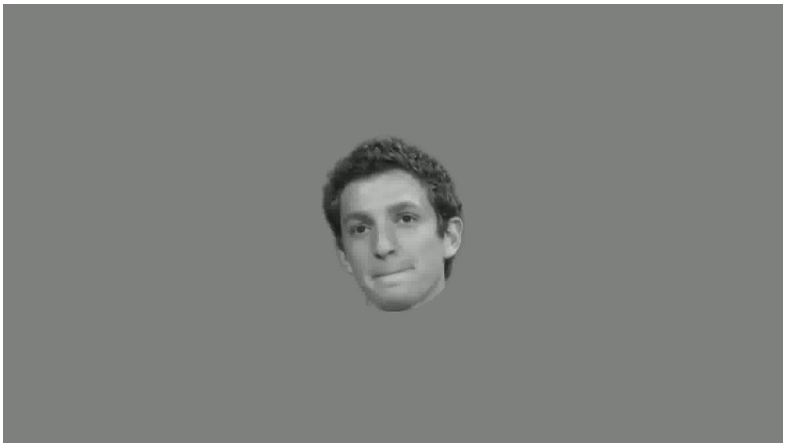

Figure 11: Stimulus representation in the paradigm

3.1.13 Strooper

The Stroop paradigm introduced by John Ridley Stroop in 1935 is a cognitive test designed to evaluate selective attention and cognitive inhibition [15]. It consists in displaying words that correspond to colors—that is, "red," "blue," "green"—printed in either congruent or incongruent ink colors.For instance, the word "red" may show up as blue ink, therefore contradicting the automatic word decoding mechanism with the regulated process of ink color choice. For participants, this conflict presents a unique challenge and provides understanding of cognitive control mechanisms.

Participants are directed to ignore the word altogether and concentrate just on the ink color, therefore suppressing natural reading tendencies. The patient has to say aloud the color of the ink the word is written in order to finish the activity. With incongruent stimuli, this work becomes particularly enlightening since they significantly influence response times and error rates relative to congruent stimuli. Said to be the "Stroop effect," these reactions vary in their degree of cognitive interference and attentional control [11].

Developmental studies extensively employ the Stroop paradigm to evaluate children's cognitive attention ranging in age from 8 to 15 [4].In clinical research, it is also a necessary tool for evaluating populations suffering with neurological and psychiatric diseases such dementia, depression, and Attention Deficit Hyperactivity Disorder (ADHD)[8]. By means of performance analysis on Stroop tasks, researchers can acquire a lot of knowledge about the fundamental deficits in cognitive flexibility and selective attention connected with these diseases.

Apart from its diagnostic value, the Stroop paradigm has been applied widely in experimental studies aiming at investigating brain correlates of cognitive control.Functional imaging studies regularly employ this paradigm to identify parts of the brain, such the prefrontal cortex and anterior cingulate cortex, active in conflict resolution and inhibitory mechanisms[3, 1]. Thanks to this flexible and sturdy construction, the Stroop paradigm is still a foundation in the study of attention, executive function, and brain adaptability.

Figure 12: Representation of a stimulus in Danish (Copenhagen version)

3.2 Definition of each inputs to be filled

This subsection explains each parameter required to run the paradigms. It is essential to recognize that not all the features discussed here are necessary for every paradigm. The correspondence between the parameters required for each paradigm will be described in the third subsection.

- **Duration between blocks:** This parameter is used to specify the time between the appearance of the new block (block of stimuli) and the end of the previous one.
- **Duration between stimuli/Duration of fixation cross:** This setting specifies the duration of the fixation cross display. Analogous to the stimulus duration, this interval is quantified in seconds and consistently applies to all fixation crosses exhibited during the paradigm.
- **Duration in seconds container stimulus:** This parameter is used to introduce the duration of the stimulus container into the auditory paradigm using artificial intelligence. Clearly, if the stimulus lasts 3 seconds and the container duration is 7 seconds, nothing will happen between the 3rd and 7th seconds.
- **Duration in seconds container cross:** Same principle as for the stimulus container, except this time it is for the fixation cross.
- **Duration of first phase, exclusion phase, transition phase:** There are three separate parameters to be input, each representing the duration of a certain phase. Nevertheless, they fundamentally fulfill the same function: to define the duration assigned to each Cyberball phase.
- **Duration of stimuli in seconds:** This parameter, as its name suggests, represents the display duration of a stimulus; all stimuli in the paradigm will be displayed for this duration, which is in seconds.
- **Instruction duration in seconds:** In the paradigm called audition, which works on the vocal cords, before each stimulus a slide indicating what the patient will have to do will appear, and this parameter can be used to indicate how long it will remain.
- **Maximum reaction time:** In Cyberball, this parameter represents the maximum time the computer can wait before returning the ball.
- **Minimum reaction time:** In Cyberball, this parameter represents the minimum time the computer must wait before returning the ball.

- **Number of blocks:** In specific paradigms, stimuli are not delivered in a strictly sequential manner, such as consistently rotating between stimulus modes and fixation crosses. Instead a block of x stimuli may be presented, with or without fixation crosses depending on the paradigm, succeeded by a fixation cross of greater duration than those within the blocks. This sequence is subsequently reiterated with one additional block. Thus, this option is used to indicate the quantity of blocks to be displayed.
- **Number of stimuli per block:** As its name suggests, this parameter is used to define the number of stimuli to be presented per block.
- **Number of training blocks:** Within the positive and negative adjective paradigm, it is feasible to incorporate training blocks, enabling the patient to rehearse prior to the commencement of the session. This parameter indicates the quantity of training blocks.
- **Select patient photo:** Selects a photo of the patient to be displayed next to his/her character during the Cyberball paradigm.
- **Patient Name:** This option designates the patient's name to be exhibited below their photograph in the Cyberball paradigm. This differs from the patient's identification, as the ID inputted in the application typically does not reflect their true name.
- **Select pre-recorded sound:** Selects an audio file containing a pre-recorded sound of the patient saying "A" for a specified duration, to be played when the patient listens in the Audition paradigm.
- **Select language:** In Strooper's paradigm, you can choose the language the patient will use during the session, so that the code can recognize what he or she is saying.
- **Random order:** If the checkbox is checked, then the stimuli will be presented in random order.
- **Select an existing paradigm:** This feature enables the selection of a file containing a list of stimuli for display. Generally, stimuli are enumerated sequentially on individual lines; however, in paradigms involving static images, other information, such as the display angle, is incorporated, formatted as "stimulus.png, 50".

- **sigma**: This parameter indicates the extent to which the values generated by a random law are dispersed around the mean. The smaller the sigma, the closer the values are to the mean; the larger the sigma, the further apart and more dispersed they are. In this case, it is always applied to the duration of the fixation cross, which is entered as a parameter and used here as the mean for the calculation.
- **Starting text:** This parameter allows for the selection of a file comprising slides that delineate the instructions for the patient, specifying their required actions during the paradigm. These slides will be presented before the commencement of the paradigm.
- **Waiting time in seconds after the Beep:** Waiting time after hearing the beep before playing a stimulus in the hearing paradigm using artificial intelligence.
- **Waiting time in seconds after the fixation cross:** Waiting time after the cross container before moving on. In the hearing paradigm using artificial intelligence.
- **Word list for the paradigm:** This parameter is only used for the scrolling words paradigm. In the input, you can indicate each word to be displayed, separating them with a comma. This parameter is used if you want to make a quick change to perform a test without having to modify a file.
- **Zoom percentage:** As its name suggests, this parameter lets you zoom in or out on the stimuli to be presented.

3.3 Parameters associated with each paradigm

This subsection outlines the mandatory input parameters required for each paradigm. While the previous subsection introduced all available parameters, this part focuses on specifying which ones are essential to launch each paradigm. Every parameter presented earlier is associated with at least one paradigm, and this mapping ensures a clear understanding of the required configurations for each specific case.

Paradigm	**Input Parameters**
Artificial intelligence image	Duration of fixation cross**,** Duration of stimuli**,** Random order**,** Select an existing paradigm, Sigma, Zoom percentage
Artificial intelligence audition	Duration in seconds container stimulus, Duration in seconds container cross, Random order, Select an existing paradigm, Sigma, Waiting time in seconds after the Beep, Waiting time in seconds after the fixation cross
Static images	Duration of fixation cross, Duration of stimuli , Random order, Select an existing paradigm, Sigma, Zoom percentage
Scrolling videos	Duration of fixation cross, Duration of stimuli , Random order, Select an existing paradigm, Sigma, Zoom percentage
Positive negative adjectives	Duration of fixation cross, Duration of stimuli, Number of blocks, Number of stimuli per block, Number of training blocks, Random order, Select an existing paradigm, Zoom percentage
Audition	Duration of fixation cross, Duration of stimuli , Instruction duration, Random order, Select an existing paradigm, Select pre-recorded sound, Sigma
Cyberball	Patient Name, Duration of first phase, exclusion phase, transition phase, Maximum reaction time, Minimum reaction time, Select patient photo

Table 1: Input parameters corresponding to each paradigm

Paradigm	Input Parameters
Emo Faces	Duration of fixation cross, Duration of stimuli , Random order, Select an existing paradigm, Sigma, Zoom percentage
Emo Voices	Duration of fixation cross, Duration of stimuli , Random order, Select an existing paradigm
Localizer	Duration between blocks, Duration of stimuli, Duration of fixation cross, Number of blocks, Number of stimuli per block, Random order, Select an existing paradigm, Zoom percentage
Scrolling Words	Duration of fixation cross, Duration of stimuli, Random order, Select an existing paradigm, Word list for the paradigm, Zoom percentage
Repetition Priming	Duration between blocks, Duration of stimuli, Duration of fixation cross, Number of blocks, Random order, Select an existing paradigm, Zoom percentage
Strooper	Duration of fixation cross, Duration of stimuli, Number of blocks, Random order, Select an existing paradigm, Select language, Sigma, Zoom percentage

Table 2: Input parameters corresponding to each paradigm

4 Creation of paradigm

4.1 Importance of the "Creation of Paradigm" Section

An essential requirement that was found at the Clinique of Saint-Luc is the capability to adapt experimental paradigms in order to cater to the special requirements of individual patients. This is the topic that is covered in the section under "Creation of Paradigm." In a therapeutic environment, every patient presents their own set of challenges, which may call for modifications to the paradigms that are currently in place in order to ensure that they continue to be effective and relevant. In order to achieve diagnostic and research goals, this adaptability is essential since it enables medical professionals to personalize paradigms without requiring a significant amount of reprogramming.

Moreover, in addition to modifying existing paradigms, it is imperative to possess the capability to formulate wholly new paradigms that markedly diverge from those now established in the application. This skill facilitates the exploration of various facets of human behavior or cerebral function that may not be encompassed by existing paradigms. By facilitating the creation of innovative experimental configurations, physicians and researchers can expand the breadth of their research and investigate new diagnostic or therapeutic methods.

At Saint-Luc, where new patients are commonly admitted, the capacity to adjust to changing circumstances is extremely important. An individual paradigm, regardless of the quality of its design, is not capable of satisfactorily meeting the different requirements of a patient group on a constant basis. It may be necessary to adapt both the visual and auditory signals in order to accommodate a patient who has hearing loss because they may experience difficulty in an environment that is mostly auditory. For the same reason, younger patients might benefit from instructions that are easier to understand or from shorter stimulus durations in order to maintain their attention. In the part under "Creation of Paradigm," clinicians are provided with the ability to efficiently execute these revisions, so ensuring that paradigms continue to be accessible, engaging, and scientifically sound.

4.2 Through the utilization of the "Audition" paradigm, demonstrating adaptability

We will examine a specific slide from the “Audition” paradigm to demonstrate the diversity of the application. This presentation has been designed to highlight two main elements: the image of a person speaking located on the left side of the slide and a yellow ear located on the right side of the slide, separated by a horizontal line in the middle. These elements were correctly positioned using PsychoPy’s Python-based features. To achieve this, the software’s inherent configuration choices for placement, scaling and overlay were exploited.

Figure 14 was reproduced manually using a rudimentary tool like Paint to assess the adaptability of the Paradigm Creation section. Despite minor variations in detail - mainly attributable to accelerated replication - there is virtually no discernible difference from Figure 13, which is a screenshot of an image presented during execution of the audition paradigm. With more time and effort, it would be almost impossible to tell the difference between the two. This shows that the Creation paradigm page can reproduce almost any visual design required for a paradigm, although it does not allow for parameters such as specific positioning.

However, the functionality of the Paradigm Creation section has certain limitations. While it now allows recording the keys pressed by the user and the time taken to press them, as well as the time from when the patient starts speaking in voice recording paradigms, it does not support creating paradigms that would perform specific actions based on a key press, like in the Cyberball paradigm, or recognize spoken content as in the Stroop paradigm. Advanced functions remain confined to the paradigms already implemented within the application.

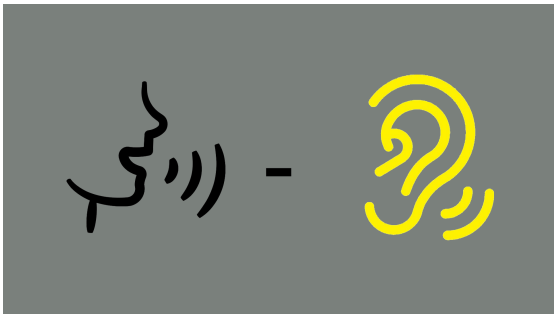

Figure 13: From Psychopy

Figure 14: From Paint

4.3 An Illustration of the Inevitability of Making Adjustments to the Paradigm

It is possible to draw an example that illustrates the importance of paradigm adaptation from the various patient profiles present at Saint-Luc. Consider a model designed to assess auditory processing through the presentation of a series of sounds in conjunction with text-based questions. This paradigm may work as intended for the vast majority of patients, but if the patient has significant hearing problems, the model may need to be modified to place greater emphasis on visual components. In this case, the "Paradigm Creation" section allows healthcare professionals to replace auditory stimuli with visual ones, ensuring that the patient continues to participate in the experience.

In addition, cognitive load is another element that can vary considerably from patient to patient. The analogous paradigm that has been adapted for children may require simpler language, fewer stimuli or longer intervals. A paradigm for adults, on the other hand, may involve elaborate instructions or extended sequences. The Paradigm Creation section enables changes to be made quickly and efficiently, ensuring that the experience is tailored to the patient's age, cognitive abilities and attention span.

4.4 In terms of clinical and research applications, the comprehensive influence

The recognition of these issues at Saint-Luc does not mean that they are particular to this clinic. Research institutes and clinical practices all throughout the world face comparable difficulties in modifying paradigms to satisfy different patient populations. The part on "Creating a Paradigm" provides a user-friendly interface for designing, altering, and implementing paradigms, therefore offering a scalable answer to this problem.

This element not only adds advanced capabilities like exact synchronizing, stimulus control, and data collecting but also greatly improves the building of visually and aurally flexible paradigms. In a therapeutic setting, these traits are absolutely crucial when dependability and accuracy are prerequisites. By changing the timing of auditory inputs, aural customization helps frameworks be adjusted to fit specific patient needs. Though they can be changed externally on the machine executing the paradigm, the application does not control volume levels; rather, it provides significant freedom.

Furthermore, the ability to save and recycle models ensures that once a customized experience is created, it may be efficiently applied for like circumstances. In clinical and research environments especially, this maintains uniformity across several use cases and saves time.

4.5 Final Thoughts

A significant amount of importance is placed on the part under "Creation of paradigm" in modern clinical and scientific contexts. At Saint-Luc, a hospital that frequently caters to a wide variety of patient needs, this flexibility ensures that paradigms may be adapted to individual requirements without sacrificing whether they are accurate or efficient. This component gives healthcare professionals the ability to change and personalize paradigms, so bridging the gap between general and specialized demands while also providing a flexible framework for the research of specific requirements.

This section provides an illustration of the relevance of this concept in the process of establishing a patient-centered approach that is suitable for the research design of clinical trials. Through the use of this method, auditory impairment, cognitive differences, and other individual patient characteristics are taken into consideration.

5 Architecture

As for the architecture of the project, I aimed to keep it as simple as possible, avoiding overly complex structures that only I could understand. To achieve this, I consulted a few people at Saint-Luc who were not particularly comfortable with computers but would be using my software, asking for their feedback on the application's structure regarding folders, input, and output files.

Although my goal was to create a simple architecture, the number of different folders (e.g., static, templates, Python files) they needed to navigate to locate the input files made the application difficult to use. To address this issue, I decided to place the input, upload and output folders outside the main folder containing all the code (HTML, CSS, Python), reducing the visible structure to just 4 folders, one of which they would never need to access.

The need to create an input folder containing a subfolder for each paradigm implemented arises from security restrictions imposed by browsers. These restrictions prevent access to the actual file path. For example, if a file is named file1.jpg, the retrieved path would appear as C:\fakepath\file1.jpg. This poses a problem for my application because, without the exact path to the file, the code cannot locate it.

For paradigms already implemented, the code explicitly defines the folder in which to look for the required files, so this does not present an issue. However, in the section dedicated to paradigm creation, I use a technique where files selected from any location on the computer are automatically copied into the Uploads folder. This ensures that all the resources required to create a new paradigm are properly organized and accessible when you want to run the new paradigm.

For output files, the format is straightforward. When a paradigm is presented to a patient, two output files are created at the start of the session. These files are filled with the same data but in two different formats: CSV and TSV. As the session progresses, these files are incrementally updated with data such as the appearance times of stimuli, the type of stimuli (e.g., stimulus or fixation cross), whether the stimulus was an image, and other related parameters.

The reason for updating the output files progressively rather than at the end of the session is to prevent data loss in case of a code crash, ensuring that the data collected before the crash remains intact. Once the experiment is completed, the code adds a new column, the duration column, to both files. This column is calculated based on the appearance times of the stimuli by simply subtracting their respective appearance times.

Afterward, a new output file in PRT format is created. This file contains parameters useful for BrainVoyager, including the number of conditions, which corresponds to the number of stimulus types. In some paradigms, there are only two stimulus types, such as stimuli and fixation crosses, while in others, like the Audition paradigm, there are several types, such as auditory, visual, vocal stimuli, and fixation crosses.
Below these parameters in the PRT file, each stimulus type is listed along with its appearance times in milliseconds and its durations.

Each of these output files is named using the format Date_PatientIdentifier _RunNumberFormat (e.g., for December 2, 2024, the file would be named 2024-12-02_342332_run1.tsv). For paradigms in which the patient's voice is recorded during execution, a folder is also created with the same identifier as the output files. For example, for December 2, 2024, the folder would be named 2024-12-02_342332_run1. This folder contains all the recordings made during the session, with each audio file named recordNbr, where Nbr is a number starting at 0 and incrementing by 1 for each subsequent recording.

	onset	trial_type	angle	reaction	stim_file	duration
1	3	Fixation	None	None	None	2020
2	2023	Stimuli	0.0	None	body19_static.jpeg	3005
3	5028	Fixation	None	None	None	2004
4	7032	Stimuli	0.0	None	body6_scr_static.jpeg	3004
5	10036	Fixation	None	None	None	2005
6	12041	Stimuli	0.0	None	face18_static.jpeg	3004
7	15045	Fixation	None	None	None	2006
8	17051	Stimuli	0.0	None	object13_static.jpeg	3003
9	20054	Fixation	None	None	None	2006
10	22060	Stimuli	0.0	None	body1_scr_static.jpeg	3003
11	25063	Fixation	None	None	None	2016
12	27079	Stimuli	0.0	None	object6_static.jpeg	3003
13	30082	Fixation	None	None	None	2005
14	32087	Stimuli	0.0	None	body4_static.jpeg	3003
15	35090	Fixation	None	None	None	2006
16	37096	Stimuli	0.0	None	body18_static.jpeg	3003
17	40099	Fixation	None	None	None	2006
18	42105	Stimuli	0.0	None	object20_scr_static.jpeg	3004
19	45109	Fixation	None	None	None	2005
20	47114	Stimuli	0.0	None	object15_scr_static.jpeg	3004
21	50118	Fixation	None	None	None	2005

Figure 15: Representation of a CSV output for the static image paradigm

```
FileVersion:        2

ResolutionOfTime:   msec

Experiment:         prt

BackgroundColor:    255 102 178
TextColor:          255 255 255

TimeCourseColor:    255 255 255
TimeCourseThick:    3
ReferenceFuncColor: 192 192 192
ReferenceFuncThick: 2

NrOfConditions:     2

Stimuli
15
  2023 5028
  7032 10036
  12041 15045
  17051 20054
  22060 25063
  27079 30082
  32087 35090
  37096 40099
  42105 45109
  47114 50118
  52123 55127
  57132 60136
  62140 65144
  67150 70153
  72159 75162
Color: 255 51 255

Fixation
16
  0 2023
```

Figure 16: Prt output obtained from the csv in figure 15

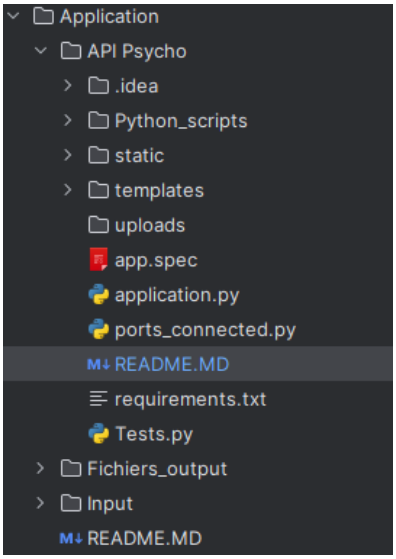

Figure 17: Application Project Tree (Development Version)

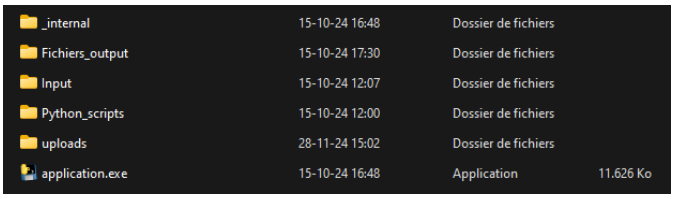

Figure 18: Application Project Tree (Portable Version)

6 Implementation

6.1 Static and Templates Folders

The static and template folders are essential for organizing and executing the visual and interactive components of the application's pages. The templates folder houses the HTML files that delineate the structure of the application's pages, whilst the static folder contains the CSS and JavaScript files required for styling and enhancing interactivity of these pages.

Instead of delving into the specifics of the HTML and CSS code, which are largely irrelevant to my thesis, I will focus on the utility of each page and the corresponding JavaScript methods that facilitate its proper operation. The screenshots in this section distinctly demonstrate the final outcome derived from the HTML and CSS, enabling us to concentrate on the fundamental mechanisms.

Every page of the application is linked to a particular JavaScript file that encompasses the functions necessary to handle the interactions and functionality pertinent to that page. Each page has a counterpart in Dutch, as I created both French and Dutch versions for all pages. These JavaScript files are crucial for the dynamic execution of paradigms and for facilitating communication between the

front-end components and the Flask server in the background.

I will begin by delineating the mechanisms and functionalities inherent to all pages, so establishing a foundational comprehension of the application's overarching structure.Following this, I will focus on the precise mechanics and distinctive aspects of each page, emphasizing their contributions to the application's overall operation.

The only functions common to all three pages of the application are related to managing overlays:

- **OpenPopup(name):** This function takes the identifier of an overlay (a div element with a hidden display attribute) and modifies it to make the overlay visible.
- **ClosePopup(name):** This function takes the identifier of an overlay and adjusts its visibility to hide it.

In addition, there are two functions shared between the Prime paradigms page and the Created paradigms page:

- **submitPort():** This function is used when performing an electrocardiogram to connect a device that records the exact moments of stimulus appearance. It retrieves the arguments entered for the port and reassigns them to other HTML elements, ensuring that the connection with the device is properly established.
- **submitPopup():** When the submit button in the overlay for changing the patient ID is clicked, this function updates the patient ID displayed in the navbar to reflect the newly entered value.

Similarly, two functions are shared between the Created paradigms page and the Creation of paradigm page:

- **highlightDuplicateTimings():** This function identifies all stimuli in the table displayed on the page (representing either the paradigm currently being created, or the paradigm selected from those already created) that are in temporal competition with other stimuli. It applies a highlight class to these lines, which changes their background color and facilitates identification of the competing stimuli.

What's more, when the user deletes a stimulus from the table, this function automatically updates the highlighting: stimuli that are no longer in competition after deletion have their highlighting removed. In this way, only conflicts that are still active are flagged, improving clarity and stimulus management in the paradigm.

- **handleCloseClick(button):** This function is used to delete the table row containing the button entered as an argument. It calls HighlightDuplicateTimings to remove the blue background on stimuli that are no longer in competition, and updates the appearance timings of stimuli after the removed stimulus so that there are no gaps in the timeline.

Each page also includes an **addEventListener** function, within which the behavior for dynamically switching the page's language is implemented. Clicking on 'NL,' 'FR,' or the toggle triggers an update to display the content in the selected language.

Now that we have explored the functions common to the various pages, let's move on to the details, structure and functionality specific to each page.

6.1.1 Prime Paradigms page

This page is the main page of my application, where all paradigms already implemented in the application (i.e., not those that can be created within the application) can be launched. The page is organized as follows:

- A sidenav on the left lists all the paradigms. Clicking on one displays text on the screen that prompts the user to fill in the parameters specific to that paradigm (list of parameters available in section 3.3 for each paradigm).
- At the top of the page there is a navigation bar:
  - On the left, there is a FR/NL toggle to switch the content to the desired language.
  - In the center, the title "Ion's PsychoPy" is displayed.
  - On the right-hand side, starting from the left, there are buttons for selecting the slides to be shown at the start of the paradigm, configuring the serial port connection, changing the patient ID, and finally there is the patient ID displayed.

The interaction between users and the paradigms is facilitated through a set of essential JavaScript functions that ensure seamless communication and execution within the application. These functions play a pivotal role in managing the

paradigms accessible through the interface. They are responsible for retrieving the parameters supplied by the user, validating these inputs, and transmitting them to the server for further processing. Despite being tailored to individual paradigms, these functions share a common structure and purpose, simplifying the transition from user input to paradigm execution while maintaining the flexibility needed to accommodate various experimental requirements.

To get started, I take advantage of a switch(section) statement, which enables the code to deal with a variety of scenarios based on the value of the section. The case in which the user clicks on one of the paradigms in the sidenav is handled by this. To be more explicit. In accordance with the paradigm that has been chosen, the relevant case is carried out, so that its particular contents, such as the parameters that must be filled in before the paradigm can be launched, are displayed.

The application currently supports thirteen paradigms, each associated with a dedicated JavaScript function. To ensure consistency and clarity, all these functions follow a standardized naming convention: submitNAMEOFPARADIGM(), where NAMEOFPARADIGM corresponds to the specific paradigm being controlled. Before launching a paradigm, users are required to complete the necessary parameters displayed on the page, ensuring proper configuration and execution.

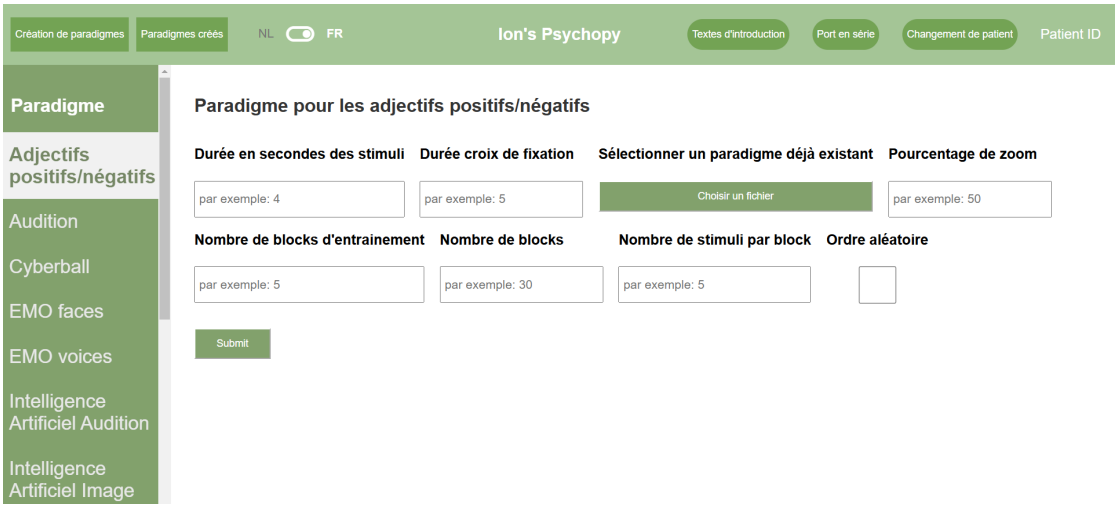

Figure 19: Representation of the Prime paradigms page with the positive/negative adjective paradigm selected

When the Submit button is pressed, an alert window is displayed to inform the user that the paradigm is about to be launched. This alert is crucial to ensure that the user knows he has pressed the submit button. In addition, it prevents the user from accidentally pressing the button several times, thus avoiding any duplication or confusion in launching the paradigm.

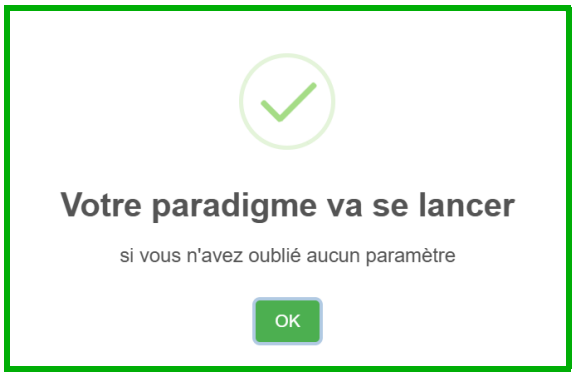

Figure 20: Alert when executing a paradigm

Activating the submit button then triggers execution of the function, which sends all the parameters required to launch the paradigm to the flask server, which in turn launches the paradigm if all parameters are satisfied. This standardized method of configuring and launching the paradigm ensures clarity of coding while promoting a simple, transparent user experience.

6.1.2 Creation of paradigm page

On this page, users can create their own paradigm. Unlike other sections of the application, here it's no longer a matter of entering parameters to run a pre-implemented paradigm, but rather of hard-coding a new paradigm, saving it and making it available for future executions.

The page is structured in a clear and functional manner. At the top, there is a navigation bar containing the title "Paradigm Creation", keeping the design clean and focused. On the left side of the navigation bar, there are two buttons: one for navigating to the Prime paradigms page and another for accessing the Created paradigms page. Below this navigation bar are six buttons, each of which opens an overlay corresponding to the specific action described on the button. Beneath these buttons, there is a table composed of five columns: Appearance, Duration, Type, Stimulus, Angle, and Remove. These columns represent the key parameters of a stimulus. However, depending on the stimulus type, not all columns will be filled. For example, for an auditory stimulus, the Angle column will display a "/" since the parameter is irrelevant. The last column, Remove, is always populated with a delete button, which allows users to easily remove rows from the table, maintaining flexibility and control over the paradigm's structure.

The six buttons, located above the table, serve the following purposes:

- **Add instructions:** This button allows users to select start and end instructions for the paradigm. Clicking it opens an overlay where users can choose two text files: the first contains the slides explaining the paradigm, which are displayed at the very beginning, and the second contains the slides shown at the end of the paradigm.
- **Adding an image:** An overlay is opened, enabling the user to select an image stimulus. The user can then define the appearance timing, specify how long the image will remain visible, and set the angle at which the image will be displayed.
- **Adding a video:** It opens a window for selecting a video stimulus. In this overlay, the user can input the appearance timing and the duration for which the video will play.
- **Adding audio:** In this overlay the user can select an auditory stimulus, where he specifies its appearance timing and duration.
- **Voice recording:** This button is used to open an overlay where the user can add a vocal recording stimulus. The overlay allows the user to set its

appearance timing and duration.

- **Fixing cross:** the last button opens an overlay dedicated to adding a fixation cross to the paradigm. The user can set its appearance timing and duration, which are then added to the table as a new row.

Once configured, the stimulus is added as a new row in the table, ensuring precise control over the timing and presentation of stimuli. With this interface, users can intuitively design their paradigms, combining different types of stimuli and precise timing parameters to fit their experimental needs.

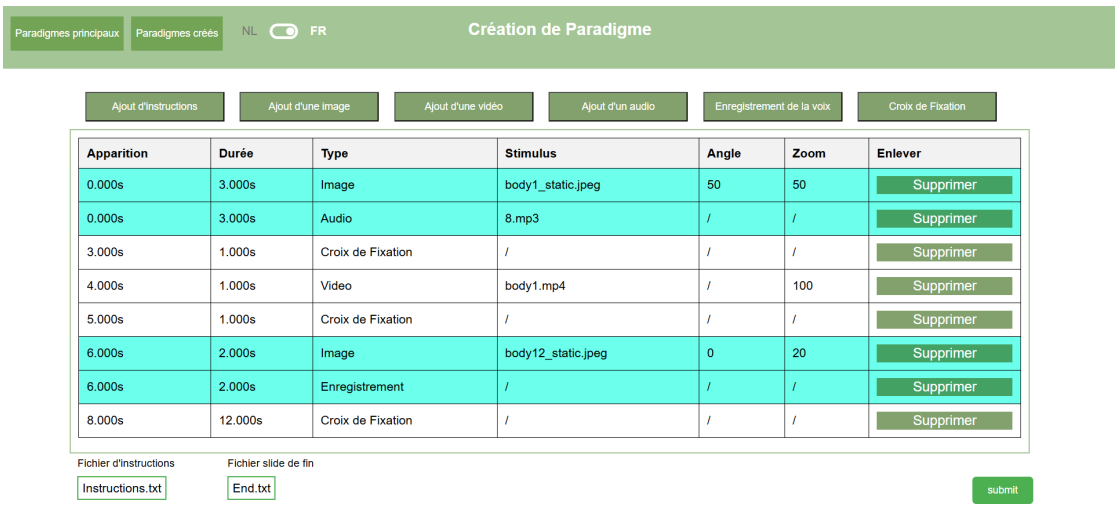

Figure 21: Creation paradigm page

Let's now explore the key functions linked to this page. These functions are designed to facilitate the configuration and management of stimuli, ensuring that each element is added seamlessly to the table. Below is an overview of their roles and how they contribute to the paradigm creation process:

- **addingFiles():** This function lets you select the introductory and final files. One showing slides explaining to the patient what's going to happen and the other showing an end slide thanking him. Once the files have been selected, they are displayed below the table.

- **submit(name):** This function retrieves the values entered when adding a stimulus to the paradigm being created. Using the name argument, it identifies the type of stimulus being added and determines which arguments are relevant. For instance, if name is "Audition," the angle variable will be set to "/," indicating that it is not applicable.

  The function then checks whether the new stimulus has the same duration as an existing stimulus at the same appearance time. If the durations match, the stimulus is added to the same row as the existing one. However, if the durations differ, an alert is displayed to inform the user that a stimulus is already active at that time, causing an overlap with mismatched durations.

  If the stimulus to be added does not share the same onset time as an existing stimulus but its addition would still create an overlap (e.g., stimulus 1 appears at 0 seconds and ends at 5 seconds, while stimulus 2 is set to appear from 3 seconds to 5 seconds), the function triggers the same alert to notify the user of the conflict.

  If no timing conflicts are detected, the function places the stimulus in the appropriate position, sorting it in ascending order relative to the Onset column.

  Once the stimulus has been added, the highlightDuplicateTimings() function is called.

- **checkingHoles():** In this function, the code searches for gaps in the timeline—for instance, if stimuli do not start appearing at 0 seconds, or if there is inactivity between the 5th and 8th seconds. Once all such gaps are identified, the function returns a list containing all the points in the timeline where gaps occur.

- **submitTable():** This function is quite simple: it is called when a user wants to submit the paradigm they have just created. The checkingHoles() function is executed, and based on the result—whether or not the timeline contains gaps—one of two overlays will appear. If gaps are detected, an overlay will propose filling all existing gaps with fixation crosses. If the user chooses not to fill these gaps with fixation crosses, they can manually add stimuli to fill the gaps. However, as long as gaps remain, the user will not be able to proceed to the next step. If no gaps are found, an overlay will appear, prompting the user to name the paradigm, allowing it to be submitted.

- **SubmitParadigme():** This function is executed when the user has named the paradigm he has just created, it takes all the data in the table and sends it to the server, which then saves the data in json format in the jsons folder, where all the paradigms created by the application are stored.

The JavaScript file also contains a variety of other functions, such as the fixation function, which operates similarly to submit(name) but only adds a fixation cross to the array, or the fillHoles function, which fills all gaps in the array with fixation crosses.

6.1.3 Created paradigms page

This page allows you to launch the paradigms you've created. Like the other two pages, the navigation bar on the left features two buttons for accessing the other pages, as well as a toggle for switching the page content between French and Dutch. At the center of the navigation bar is the title 'Paradigms Created.' On the right, similar to the Prime Paradigms page, are two buttons: one for configuring the serial port and/or modifying the trigger used to start the paradigm, and another for updating the patient identifier. The current patient identifier is displayed to the right of these buttons.

This page is something of a visual blend of the other two pages. On the left, there is a sidenav listing all the paradigms you've created, while in the center, a frame displays a table showing the stimuli that belong to the selected paradigm. Below this frame, there is a button to randomize the order in which the stimuli will be presented, and on the right, a submit button allows you to launch the selected paradigm.

The majority of the functionality on this page relies on functions already present and explained in the other pages. Therefore, the only two functions specific to this page and worth detailing are those that manage the dynamic update of the sidenav and the retrieval and display of stimuli for the selected paradigm.

The first one is the **addEventListener** function that dynamically updates the sidenav to display all the paradigms created by the user. When the page loads, it sends a request to the flask server to retrieve the list of available JSON files, which represent the paradigms. The sidenav is reset to keep only its header ('Paradigme'), and for each file returned, a clickable link is added. Clicking on a link triggers the get_table(file) function to load the table of stimuli for the selected paradigm. This ensures that the sidenav reflects any newly created paradigms in real-time, providing a seamless and interactive user experience. If an error occurs during the fetch process, it is logged to the console for debugging purposes.

The **get_table(file_name)** function retrieves and displays the data for a selected paradigm, facilitated by the Flask server. It updates the table with rows for parameters such as appearance time, duration, and type, and includes a 'Supprimer' button to delete rows. The function also calls highlightDuplicateTimings() to visually indicate overlapping stimuli.

Figure 22: Created paradigms page

6.2 Python Files

6.2.1 Parent class

As I was writing Python code to create paradigms, I realized that certain functions were redundant between different files. To optimize my code and improve its maintainability, I decided to create a mother class grouping all the functions common to my different files. In what follows, I'll detail each of the functions implemented in this parent class, explaining their role and usefulness in the context of my paradigms.

- **launching_texts:** This function enables neurologists to display a series of slides with instructions explaining what the patient must do throughout the experiment. It takes four arguments: the first is the PsychoPy window, which is the window the patient will see once the paradigm is launched; the second is a list of texts to be displayed sequentially on the screen when the patient or the neurologist/technologist presses a key; the third is the text alignment, which by default is set to the left but can be centered in certain paradigms based on specific neurologist requests. Finally, the fourth argument is the trigger, which is a crucial parameter as it defines the character that will initiate the experiment and is not used for transitioning between instruction slides.

- **the_end:** This function is used to display a final slide at the end of the experiment to indicate that the experiment is over, to thank the user, and to inform them that they will be spoken to in a few seconds. It takes only one argument: the PsychoPy window. Variants of this function exist to display different end screens depending on the paradigm that calls it.

- **file_init:** In this function, we initialize output files in CSV and TSV formats, where we define the column names of interest. For example: onset, stimuli, trial_type, response, time_before_starting_to_answer.

- **preprocessing_tsv_csv:** This function first checks whether the "File_outputs" folder exists, and if not, it creates it at the specified destination. It then retrieves the current date in Year-Month-Day format and checks whether any other files exist with a name matching the current date followed by the patient's ID and _run. If such files are found, the function retrieves the filename with the highest number following _run and increments it by 1. It then creates two output files in CSV and TSV formats, named using the current date, the patient ID, and the incremented run number. These files

will serve as outputs for the current experiment. The function takes only one argument: the patient ID.

- **write_tsv_csv:** As its name suggests, this function writes data to the CSV and TSV output files previously created by the preprocessing_tsv_csv function. It takes three arguments: the first is the CSV and TSV files to be written to, the second is the data to be written, and the third is the line number to which the data will be written.

- **writting_prt:** This function, while short, has a specific purpose: it instantiates an object of the writtingprt class to process results stored in the CSV file and make them compatible with BrainVoyager. Specifically, it generates a third output file based on one of the other two, enabling the results to be used in BrainVoyager. The details of its utility with BrainVoyager will be explained in Section 9. The function takes only one argument: the name of the CSV file.

- **inputs_texts:** This function retrieves all the texts that the neurologist launching a paradigm wants to display as instructions within the paradigm. It takes a single argument: the path to the file containing the texts. It returns these texts as a list.

- **proper_waitkey:** This function launch a loop of infinite duration, during which we check each time that the trigger to launch the paradgim has not been pressed. Once the trigger has been pressed, you exit the loop and the paradigm is launched. This function takes just one argument, and that is the trigger.

- **send_character:** This function is only used when an electrocardiogram is being employed, and it sends a signal to a small box via a serial port whenever a stimulus is displayed on the screen, making it possible to precisely determine the timing of stimulus onset in order to ensure synchronization with the electrocardiogram machine. It therefore takes into account the port through which the device is connected and the baudrate to ensure that the serial connection is successful.

- **adding_duration:** This function simply calls the adding_duration1 function twice; for each output file of a different type.

- **adding_duration1:** In this function, we open the two output files, CSV and TSV, and transform all appearance times from seconds into milliseconds. Once this transformation is complete, we add a stimulus duration column by calculating the difference between the onset times of these stimuli.

6.2.2 Psychopy_everything

In this subsection, we will discuss the Psychopy_everything(Parent) class, which serves as a cornerstone for enabling users to launch their own paradigms once they have created them. This class was designed with versatility in mind, ensuring that it can handle stimuli of various types and, in certain cases, display them simultaneously.

I have chosen to focus on this class because it provides a general framework capable of managing all types of stimuli, making it representative of the other Python classes linked to specific paradigms. Describing the functions of each individual class for specific paradigms would require an extensive number of pages and would not add significant value, as most of their functionality is largely summarized by the features and flexibility of the Psychopy_everything class.

The primary goal of this class is to provide a flexible and adaptable framework capable of accommodating a wide range of paradigms, even if these paradigms differ significantly in structure or functionality. By maintaining an open architecture, the class allows for the seamless integration of diverse experimental designs while preserving the essential features required for their execution.

Creating this class posed a number of challenges, particularly in ensuring compatibility between stimuli with different properties, such as timing, presentation modes, or types of interactions. Despite these complexities, the class remains robust and general enough to support paradigms that may not resemble each other at all, ensuring reliable performance regardless of the experimental demands.
In the following parts, I will delve into the specific functions within this class, detailing their purpose and how they contribute to the overall goal of making paradigm execution as efficient and user-friendly as possible.

We will begin with the ___init___ function, as it plays a crucial role in the class by instantiating the core variables required for the proper functioning of the paradigm. These variables serve as the foundation for managing stimuli, timing, and data storage, and they are closely interconnected to ensure the seamless execution of the paradigm. Let us now explore each variable and its specific role within this framework.

The initialization process begins with the creation of the PsychoPy window where all stimuli will be presented during the paradigm. This window serves as the central interface for delivering the visual and auditory components of the experiment.

Next, several key variables are set up to manage the stimuli before they are displayed. Specifically, there are four dedicated lists to hold static image stimuli, video stimuli, auditory stimuli, recording stimuli and fixation cross stimuli. These lists act as temporary storage, ensuring that each stimulus is properly organized and ready for presentation. One might wonder why there's a list for voice recordings, given that there's no stimulus to show. The purpose of having a list for this type of stimulus is to have information such as how long the patient should be recorded for, or from what point onwards.

A tracking variable is also initialized to monitor the current position within the list of stimuli. This tracker ensures that the application knows which stimulus to present next and maintains the correct sequence of events. Additionally, a pointer variable is created to reference the specific data of the current stimulus being presented. This variable allows the program to retrieve detailed information about the stimulus, such as its appearance timing, duration, angle (if it is an image), and other parameters critical to its presentation.

For efficiency, the fixation cross is instantiated during initialization, eliminating the need to recreate it each time it is needed. This saves processing time and ensures consistency in its presentation throughout the paradigm.

Another important variable points to the folder containing all resources related to the stimuli. This ensures that the program can easily access the required files, such as images, videos, and audio, without unnecessary overhead. Variables related to microphone initialization are also set up, including parameters for configuring and preparing the microphone for any auditory recording tasks.

Finally, the initialization includes the creation of a global clock. This clock is essential for measuring the duration of stimuli, enabling precise timing for transitions between stimuli and ensuring that the paradigm adheres to its intended schedule.

Having covered the initialization process, we now turn our attention to the core functions of the class. These functions are fundamental to the proper execution of the paradigm, handling tasks such as stimulus presentation, timing management, and data collection. Below is a list of the most important functions, along with a brief explanation of their purpose and role within the class.

- **show_croix(self):** Displays a fixation cross on the screen.
- **show_image(self):** Handles the presentation of an image stimulus.
- **show_video(self):** As its name suggests, this function is responsible for displaying a video stimulus during the paradigm.
- **show_audio(self, need_image=True):** This function is used to play an audio stimulus. However, unlike other stimulus-related functions, it takes an additional argument, need_image. This argument determines whether an accompanying image should be displayed when the audio stimulus is presented. For instance, if the audio is presented independently, need_image will be set to True, and an image will be added. Conversely, if another image stimulus is already being shown simultaneously, need_image will be set to False, ensuring that no additional image is displayed.
- **show_enregistrement(self, need_image=True):** This function manages a vocal recording stimulus. Like the show_audio function, it incorporates the need_image parameter to determine if an accompanying image should be presented. When the recorded stimulus is provided independently, need_image will be assigned a value of True, resulting in the addition of an image. However, if another visual stimulus is concurrently displayed, need_image will be set to False, preventing any extra image from overlapping the recording process.
- **show_good_type(self, type):** This function takes a single argument, type, which specifies the type of stimulus to be presented. Based on this argument, the function identifies the appropriate list corresponding to the stimulus type, retrieves the first element from that list, removes it to maintain proper sequencing, and then presents the stimulus. This ensures that stimuli are displayed in the correct order and that the list remains updated after each presentation.

- **show_multiple_types (self, types):** This function operates similarly to show_good_type, but instead of taking a single type argument, it uses types, allowing multiple stimulus types to be presented simultaneously. For each type specified in the types argument, the function retrieves and presents the corresponding stimulus. While all stimuli are displayed at the same time, the use of threads depends on the specific combination of stimulus types being presented. Threads are only employed when certain types require independent handling to ensure proper synchronization and execution.

- **lancement(self):** This function is responsible for calling the appropriate functions to display stimuli. It examines the current element in the list of stimuli to be presented and determines whether it contains multiple stimuli. If multiple stimuli are detected, it calls show_multiple_types to handle their simultaneous presentation. Otherwise, it defaults to show_good_type to display a single stimulus. This ensures that the correct method is used depending on the complexity of the stimuli configuration.

- **preprocess(self):** This function is responsible for pre-processing the stimuli, organizing them into their respective lists based on stimulus type, and maintaining a sequence that dictates the order in which they should be presented. For certain types of stimuli, where the necessary data is readily available and processing them in advance is not overly resource-intensive, the stimuli are created immediately. For instance, all image stimuli are pre-created during this step, ensuring that they are ready for presentation without requiring additional processing at runtime. This approach optimizes performance by reducing delays during the paradigm execution, particularly for stimuli that require minimal computational overhead to prepare. If the option to randomize the order of stimulus presentation is enabled, the application checks for competing stimuli. If there are no competing stimuli, it shuffles each list of stimulus type to ensure a randomized sequence. However, if competing stimuli are present, the application does not shuffle the lists to maintain synchronization between the stimuli, as managing randomness with synchronization in such scenarios becomes too complex.

7 Package Versions for the Proper Functioning of the Application

Using the correct versions of packages is essential for the stability and proper functioning of the application. Each package plays a specific role in managing the paradigms, handling data, or ensuring seamless communication between the front end and the experimental scripts. Additionally, it is crucial that the application runs in a Python 3.10 environment, as the compatibility of several packages depends on this version. The following provides an overview of the key packages and their importance to the application.

- **dukpy (0.2.3)**
  - **Role:** Executes JavaScript code within Python.
  - **Importance:** Used in scenarios where dynamic handling of JavaScript is required server-side, such as processing logic from the user interface directly within Flask routes.
- **Flask (2.3.3)**
  - **Role:** A web framework to manage application routes and APIs.
  - **Importance:** Central to the application, it manages user inputs, handles file uploads, and communicates parameters (e.g., duration, stimuli) to experimental Python scripts.
- **fonttools (4.53.1)**
  - **Role:** Handles font manipulation.
  - **Importance:** Supports the rendering of text in paradigms that involve customized fonts or specific stylistic requirements, ensuring consistency in the display of stimuli.
- **imageio (2.35.1)** and **imageio-ffmpeg (0.5.1)**
  - **Role:** Handle image and video processing.
  - **Importance:** Used in paradigms involving visual stimuli to load and display images or videos efficiently.
- **PsychoPy (2024.2.1)**
  - **Role:** The main library for creating and executing paradigms.

  - **Importance:** Handles visual and auditory stimuli, experimental timing, and user responses. It is the backbone of experimental execution.

- **sounddevice (0.5.0)** and **soundfile (0.12.1)**
  - **Role:** Manage audio input/output.
  - **Importance:** Crucial for paradigms like "Audition" that involve recording or playing back audio, enabling synchronization with stimuli.

- **pygame (2.6.0)**
  - **Role:** Provides multimedia capabilities.
  - **Importance:** Used for tasks such as managing audio playback in paradigms, ensuring precise delivery of auditory stimuli.

- **numpy (1.22.4)**
  - **Role:** Numerical data manipulation.
  - **Importance:** Handles calculations for experimental logic, such as randomizing stimuli or generating Gaussian distributions for timing variability.

- **pandas (2.2.2)**
  - **Role:** Tabular data manipulation.
  - **Importance:** Processes results from paradigms, such as reading and writing data files in CSV/TSV formats for analysis.

- **Pillow (10.4.0)**
  - **Role:** Image manipulation.
  - **Importance:** Supports tasks like resizing and rendering images for paradigms that involve visual stimuli.

- **SpeechRecognition (3.10.4)**
  - **Role:** Provides voice recognition.
  - **Importance:** Integrated for experiments requiring vocal inputs, such as categorizing stimuli based on spoken responses.

- **waitress (3.0.0)**
  - **Role:** A WSGI server for deploying Flask applications.
  - **Importance:** Ensures stable and efficient performance in production environments.
- **Werkzeug (3.0.2)**
  - **Role:** Core WSGI utilities used by Flask.
  - **Importance:** Manages HTTP requests and responses, ensuring smooth communication between the front end and back end.

8 Technical challenges

8.1 Start-up and installation

When I started, I was completely new to Psychopy and to the concepts of psychological paradigms. Getting started was challenging, particularly because of the numerous dependencies required by this software at the code level to make it functional. Once I managed to get it running, it was relatively straightforward to display basic stimuli that required minimal parameters (for example, displaying an image without having to manage its zoom, angle, or position on the screen). However, as mentioned, the large number of dependencies made it difficult to execute the code on different computers without spending significant time installing all the dependencies and ensuring their compatibility.

To remedy this, I installed my application on one of the clinic's computers connected to the scanner, which enabled us to run and test it without any problems. I also installed it on Ron's computer so that he could use it with his patients in Copenhagen. However, this solution was not viable in the long term, as it limited the number of computers on which the application could be used. Each installation required someone with advanced computer skills to ensure that it was correctly configured. What's more, the people who had to run the application had to go through the code editor to launch it, which meant he could easily add a character to my code unintentionally, rendering it completely unusable.

I then turned to PyInstaller, a tool that transforms Python scripts into executables by gathering all the necessary dependencies either directly in the executable or in a folder next to it, depending on the chosen configuration. In theory, this approach seemed ideal. I tested it on simple code using PsychoPy, and it worked well, as it did for the front end of the application using flask. However, when I

combined everything into a single executable, I couldn't get the Flask and PsychoPy components to work simultaneously. Although I tried various approaches, like different sockets, threads, or sub-processes, I struggled with this problem for quite some time before finding a solution.

The best solution, given the complexity of PsychoPy's dependencies, was to create separate executables for each code module that used it. This approach has one particular advantage: compiling PsychoPy-based Python code into an executable with PyInstaller is a time-consuming process (on my computer, it often takes around an hour and a half). By separating the executables, modifying a specific paradigm no longer requires recompiling the whole application - only the code for that paradigm needs to be recompiled. This considerably simplifies code maintenance and reduces the time wasted during development.

8.2 In paradigms

In the scrolling videos paradigme, we encountered the same issue that the clinic had previously experienced with E-Prime: as the experiment progressed, the videos became increasingly less fluid, eventually appearing as static images rather than moving visuals. To address this, I conducted several trial-and-error attempts before identifying the root cause of this loss of fluidity. Previously, my code did not manage resource release effectively. Although Python typically handles resource management automatically, this is not the case when displaying videos through PsychoPy. Consequently, I implemented a mechanism to ensure that, at the end of each video, all associated resources were released and the memory was properly freed.

Unfortunately, this was not the only issue with this paradigm. When I attempted to display the stimuli provided by the clinic, which featured a blue background, the application crashed, preventing the experiment from being completed. Despite making several modifications to my code, the problem persisted, and PsychoPy continued to crash. To address this, I re-encoded all the videos that I had received with a blue background, ensuring they were in the correct format. After this re-encoding, no further errors occurred with the scrolling videos paradigm.

In addition the audition paradigm posed a considerable technical challenge due to its complexity. It was the first instance where a paradigm required the simultaneous execution of multiple actions. To address this complexity, I implemented the use of threads to enable, for example, the broadcasting of a sound into the patient's ear while simultaneously recording their voice. Precise synchronization of thread startup and termination was essential to ensure the desired behavior,

both in terms of patient interaction—such as voice recording coupled with audio file playback—and in terms of the output file. Without proper management of the main thread to wait for the completion of the secondary thread running in parallel, encoding results were occasionally incomplete, with crucial data lost when the secondary thread failed to complete synchronously.

8.3 Continuous adaptation throughout the thesis

Throughout the thesis, the project required the continuous incorporation of new paradigms and functionalities to meet the particular requirements of the clinic. This mission required frequent, precise and sometimes urgent modifications, usually executed at late hours, in order to guarantee the application's full functionality. The constant integration of these elements enabled the project to evolve, underlining the challenge of aligning technological advances with ever-changing clinical requirements, right up to the end of my thesis.

9 Usefulness of output files

As a student of computer science, my expertise lies mainly in software development, not functional neuroimaging analysis. This section draws on a discussion with Laurence Dricot, an expert in this field, to explain how the output files generated by my paradigms are integrated and used in BrainVoyager. This software, widely recognized for its analysis of fMRI data, plays a central role in the interpretation of brain activations. Although some of the technical details may escape me, this section aims to illustrate how my programming skills fit in with neuroscientific tools.

The paradigms established in my application generate output files, in CSV and TSV formats, from which a PRT file compatible with BrainVoyager is produced. This document comprises critical information:

- Exact timing of the stimuli presented during the paradigms.
- Type of stimuli (faces, words, etc.).
- Temporal markers denoting fixation crosses.

These information is essential for precise analysis in BrainVoyager. Incorrect timings or absent conditions can add substantial mistakes into analytical models, hence diminishing the reliability of the conclusions.

BrainVoyager utilizes the output files to construct a design matrix, serving as a temporal representation of the experimental conditions linked with the fMRI data. This matrix denotes the timing of each condition's presentation, facilitating the correlation of measured signals with stimuli. For instance, if a visual stimulus is displayed for a duration of 3 to 7 seconds, succeeded by a rest interval from 7 to 12 seconds, this data is included in the design matrix. This allows BrainVoyager to accurately correlate the neural activations recorded during these times.

The design matrix is subsequently employed in the implementation of a General Linear Model (GLM), which functions as a fundamental instrument for the analysis of fMRI data. The GLM estimates beta coefficients by modeling the link between the timing of experimental conditions and the recorded neural signals. These coefficients measure the degree to which each voxel in the brain reacts to particular experimental conditions. A voxel that is highly responsive to a "face" stimulus would demonstrate a significant beta coefficient associated with that condition.

The GLM analysis enables researchers to visualize brain activations as statistical maps, emphasizing regions significantly activated by specific stimuli. This methodology enables researchers to create statistical maps that emphasize brain areas strongly engaged in particular experimental settings. By analyzing these activations, researchers can gain insights into the contributions of various brain regions to cognitive, emotional, or sensory processing. Such discoveries are crucial for enhancing our understanding of the brain's functional organization and its reactions to certain stimuli.

Integrating the output files into BrainVoyager's analytical framework allows for the accurate identification of neural correlates linked to experimental activities. This integration highlights the significance of software development in connecting computational tools with neuroscientific research, enabling data-driven insights into brain function.

10 Tests

10.1 Initial Testing and Focus on Flask

Initial testing of the application consisted primarily of human analysis for experimental paradigms. This was done in order to ensure that the application was functioning properly. That is why with Ron, once a paradigm had been developed, we made certain that it functioned appropriately without having to perform tests against the code that was provided. Because the majority of the code needed to be ready for usage in a short amount of time, this technique was required. I did not begin writing tests until after these hectic moments had passed, and the most of them focused on the Flask component of the application.

10.2 Testing Methods: Routes, JSON Management, and Paradigms

I built a set of tests for the application.py file, which is responsible for centralizing everything that occurs within the program. This includes data processing, file saving, and the transmission of variables to Python files that launch PsychoPy to show paradigms based on the variables that have been received. A total of 1070 lines of code are included in these tests, which cover 88 percent of the application.py file.

The uncovered lines are mostly associated with instances in which I did not carry out the necessary checks to see whether the function successfully raises an error when an issue arises while it is being executed. I will provide an explanation of the thought process that led to this choice in the following paragraphs. One specific function was not tested during the development process, as testing it was deprioritized in favor of other components. Instead, I manually verified its behavior by ensuring that it accurately generates the files expected to be present in the designated folder. This verification confirmed that the function operates as intended and reliably fulfills its purpose.

```
PS C:\Users\coren\Documents\Memoire\Application\API Psycho> coverage report -m
Name            Stmts   Miss  Cover   Missing
---------------------------------------------
Tests.py          266     23    91%   23-24, 144-158, 225-238, 518-530, 622-641, 1017-1036, 1062
application.py    392     46    88%   18-26, 65-75, 101-102, 124-125, 168-169, 207-208, 278-279, 325-326, 374-375, 421-422, 471-472, 519-520, 611-612, 660-661, 713-715
---------------------------------------------
TOTAL             658     69    90%
```

Figure 23: Coverage report

The first thing I did was use the Flask application to test all of the route redirections. To do this, I first determined whether or not accessing a route resulted in an error, and if it did not, I checked to see if the page that was returned contained an element that was peculiar to that particular page.

In the following step, I examined the management of JSON files, which are files that reflect paradigms that were manually created through the paradigm creation page. In order to verify this, I made a JSON file that included only a small amount of data and put it in the area that was anticipated for its storage. After the creation of the JSON file, I confirmed that my test file was included in the list by calling the method that was responsible for getting all of the JSON files that had been stored. I then erased the test file in order to keep the environment as clean as possible subsequently.

Finally, for each paradigm already implemented in the application, I test via a json that I create manually the sending of it via a request for this paradigm. Ff there are no errors then I receive a response containing a status of 200 (meaning that it worked) and a json message 'status': 'success', 'message': 'Data received and script executed'.

10.3 Simulating External Scripts and Error Handling

I utilized mock for the purpose of conducting tests that required verification of the execution of an external Python script through the use of subprocess.run. Thanks to this, I was able to simulate the script call without really executing it, which was an essential step in isolating the test. While using mock, I checked to make sure that the command and arguments that were generated were accurate. In addition, I simulated both successful and unsuccessful cases in order to verify that the application replied effectively, independent of the state or content of the external script. With mock, I was able to make certain that the arguments that were supplied to the subprocess.run function were identical to those that were necessary for subsequent paradigm executions.

After I had established that the functions that were responsible for delivering parameters for paradigm execution were functioning appropriately, I proceeded to test scenarios that were likely to result in mistakes. By way of illustration, I either utilized erroneous arguments or failed to match the number of arguments that were required by the script with the number that was present in the JSON file. In order to guarantee optimal error handling across all of the different paradigms, I devised a series of tests. On the other hand, due to the fact that these functions operate in a comparable manner, failure cases were tested on some of the functions but not

all of them. This partially explains why the file has a line coverage of less than one hundred percent.

10.4 Paradigm validation with analysis of output files

Subsequent to the development of each paradigm, Ron and I conducted a thorough validation by examining the produced output files. The objective of this essential phase was to verify that each paradigm operated entirely as anticipated, without discrepancies or errors. Upon validation of a paradigm, it was deemed dependable and suitable for application, precisely aligning with the specified requirements.

This method guaranteed that paradigms conformed to clinical criteria immediately upon validation, while also being changed without sacrificing their integrity or functionality. This stringent validation technique ensured the efficacy and precision of the paradigms throughout the thesis.

11 Limitations and future work

An important constraint discovered during this project concerns the way PsychoPy presents stimuli and manages resources. Certain types of stimuli, in particular videos, have delays associated with loading in real time before being displayed, as mentioned in the section on technical challenges. These delays, although observable, vary according to the type of stimulus processed. This variation is due to the complexity of the calculations required to process these stimuli. For example, basic visual stimuli such as images require a minimal amount of system resources, whereas videos require smooth playback, which places greater demands on the processor and RAM. Voice recordings, on the other hand, do not require a pre-loading process, but do require real-time capture of the patient's vocal reactions. Despite this, voice recordings are also influenced by other types of technical constraints. This feature adds a further technical layer to paradigms requiring simultaneous control of recording and stimuli, which must be meticulously synchronized with each other. The most important point in the clinical context for which this application is intended is not the perfect synchronization of the stimuli at the level of the parameters initially entered, but rather the tool's ability to accurately record the moments of onset, disappearance and, in the case of voice recordings, the precise moment when the patient begins to speak. This feature guarantees that the data generated will remain usable and reliable from a scientific point of view, a considerable advantage for researchers and doctors alike.

A series of tests designed to evaluate the appearance times of different types of stimuli (images, videos, sounds and voice recordings) was carried out by myself in order to illustrate these limitations while highlighting the program's resilience. The aim of these tests was to gain a better understanding of how experimental paradigms are affected by technical constraints, and to evaluate the application's performance objectively. I was able to recognize specific patterns by recording when each stimulus appeared on a number of trials (the stimuli all had a basic duration of 1 second apart from the audio recordings which had a duration of 0.55 seconds). By way of illustration, the videos showed significant gaps, frequently exceeding 0.5 seconds, in contrast to the images, which appeared virtually instantaneously with a slight average delay at the moment of appearance. As far as voice recording is concerned, the process mainly depends on the patient's response and vocal threshold; nevertheless, simultaneous control of response capture and recording guarantees total reliability of temporal data. This type of review is essential to provide a comprehensive assessment of the program's capabilities, while ensuring that its use is transparent and in line with scientific expectations. The following table summarizes the average time losses observed for each stimulus type, and shows the onset times recorded during one of the test runs.

Image	**Video**	**Audio**	**Record**
0,00001	0,29758	0,01465	0,26018
1,00099	1,87547	0,57164	1,57337
2,00205	3,36545	1,12682	2,84649
3,00300	4,84865	1,69142	4,12491
4,00390	6,33280	2,24594	5,42680
5,00472	7,81475	2,80080	6,71666
6,00559	9,29937	3,35539	7,99558
7,00645	10,78492	3,91020	9,30108
8,00735	12,26637	4,47594	10,58448
9,00818	13,78676	5,03042	11,86766

Table 3: Time of appearance of each stimulus for each type

Image	**Video**	**Audio**	**Record**
0,00091	0,49880	0,00599	0,28972

Table 4: Average time lost per type of stimulus

According to the results, the delays experienced by sounds and images are almost minor, with an average of 0.00599 seconds and 0.00091 seconds, respectively. In the context of the proposed paradigms, these insignificant losses are not perceptible and have no effect on the results obtained. On the other hand, video loading and display times were significantly longer, averaging 0.49880 seconds. Voice recordings, on the other hand, are not subject to a loading procedure, but their processing is based on real-time recording of patient responses. Thanks to its well-specified sound threshold, the program is able to record precisely when the patient begins to speak.

These delays are primarily caused by PsychoPy's win.flip() function, which updates the displayed window. A minor contribution may also stem from the optimization level of the current code. While further optimization of the existing code could help reduce these delays, it is unlikely to eliminate them entirely.

These findings demonstrate that, despite the fact that the program has some technological constraints, it is perfectly matched to the primary goals that it aims to achieve; its conceptual effectiveness and robustness in situations where data dependability and precision are vital are demonstrated by the fact that it is able to reliably document the timing of stimuli and capture reactions in real time, even when the conditions are demanding.

The program also has two other limitations. For the moment, in the paradigm creation section, we can not randomize the list of stimuli to be shown if we have created a paradigm involving competing stimuli. To fix this, we would have to record a tracking of the competing stimuli that have similar timings to see if we can randomize them, and randomize on the side the stimuli that are not competing. Second, while the output files are specifically tailored for BrainVoyager, they might not be directly compatible with other analysis software. To address this, a Python script could be developed to convert the CSV output into the desired format for other tools, ensuring broader compatibility.

12 Github repository

The complete code is accessible to the public on GitHub at https://github.com/Corentin-student/Psychopy-Scanner. The repository comprises two branches: For_py_path and For_exe_path. The distinction between these branches resides in the access paths delineated in application.py for particular directories, which are crucial for the proper functioning of the code based on its usage format—either as Python scripts in an IDE or as generated .exe files.

This branch contains the code configured for development and execution within an integrated development environment (IDE). The direct accessibility of the source code makes it optimal for debugging or modifying the application.

For_exe_path: This branch is designed for utilization subsequent to the conversion of the code into executable files (.exe). It encompasses all requisite modifications to path configurations to enable the application to operate smoothly in a standalone mode, independent of an IDE.

The For_exe_path version has been compiled into several executables and deployed at the Institute of Neurosciences and Clinique Saint-Luc. These executables are intended for various functions and can be readily installed on other computers by transferring the files using a USB drive. This mobility facilitates adoption in clinical and research settings, obviating the necessity for intricate installations or Python configurations.

Both branches provide extensive documentation on the requisite configurations and procedures for setting up and using the application in its respective formats. The repository offers all necessary resources for both development and portable application use.

13 Conclusion

This dissertation proposes an innovative response to the challenges associated with the design and use of psychological paradigms in clinical and research environments. The application developed considerably simplifies these processes, thanks to a user-friendly interface, modular architecture and efficient integration with analysis tools such as BrainVoyager. It stands out for its ability to adapt paradigms to specific needs, whether this involves modifying visual or auditory stimuli, recording vocal responses in real time, or precisely synchronizing data with external devices.

The application's flexibility and robustness have been demonstrated in a variety of contexts, from Clinique of Saint-Luc to international collaborations. The paradigms included, such as the Stroop or Cyberball, enable complex aspects of cognitive and emotional functions to be explored, while offering a high degree of parameter customization to suit the specificities of participants or experiments. Integration with BrainVoyager enhances this capability by providing accurate and compatible output files, facilitating analysis of correlations between experimental conditions and brain activations.

Despite these successes, the project also highlights areas for improvement. For example, the delays observed in loading videos and the limitations of stimulus synchronization represent challenges to be resolved. Although these aspects do not affect the quality of the data collected, they open the way to future optimizations. Furthermore, the addition of more complex paradigms and the integration of interactive visualizations could further broaden the application's impact.

In short, this project represents a significant step forward in the use of technology in the human sciences. By reconciling simplicity of use, adaptability and scientific rigor, it lays the foundations for a promising tool, capable of meeting the requirements of clinicians and researchers while fostering the emergence of new perspectives on cognitive and emotional mechanisms. This multidisciplinary approach opens up new opportunities for the exploration and application of psychological paradigms, strengthening the dialogue between technological innovation and scientific research.

Bibliography

[1] Marie T. Banich, Michael P. Milham, Ruth Atchley, et al. Prefrontal regions play a predominant role in imposing an attentional 'set': Evidence from fmri studies of the stroop task. *Cognitive Brain Research*, 10(1-2), 2000.

[2] Rainer Banse, Klaus R. Scherer, Marcello Mortillaro, and Didier Grandjean. Introducing the geneva multimodal expression corpus for experimental research on emotion perception. *Emotion*, 12(5), November 2011.

[3] Cameron S. Carter, Mark Mintun, and Jonathan D. Cohen. Interference and facilitation effects during selective attention: An h215o pet study of stroop task performance. *NeuroImage*, 2(4), 1995.

[4] Charles J. Golden. *The Stroop Color and Word Test: A manual for clinical and experimental uses*. Stoelting Co., 1978.

[5] Thomas S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, 1962.

[6] Thomas S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, 1962.

[7] Thomas S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, 1962.

[8] Marit M. Lansbergen, J. Leon Kenemans, and Herman van Engeland. Stroop interference and attention-deficit/hyperactivity disorder: A review and meta-analysis. *Neuropsychology*, 21(2), 2007.

[9] Larousse. Paradigme. `https://www.larousse.fr/dictionnaires/francais/paradigme/57869`. Consulté le 28 décembre 2024.

[10] Thomas Hardy Leahey. *A History of Psychology: From Antiquity to Modernity*. Routledge, 8th edition, 2018.

[11] Colin M MacLeod. Half a century of research on the stroop effect: An integrative review. *Psychological Bulletin*, 109(2):163–203, 1991.

[12] Abraham H. Maslow. A theory of human motivation. *Psychological Review*, 50(4), 1943.

[13] G. A. Miller and J. Keller. Psychology and neuroscience: Making peace. *Current Directions in Psychological Science*, 9(6), 2000.

[14] B. Douglas Slife and Richard N. Williams. *What's Behind the Research?: Discovering Hidden Assumptions in the Behavioral Sciences*. SAGE Publications, 1995.

[15] John Ridley Stroop. Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, 18(6), 1935.

Appendix

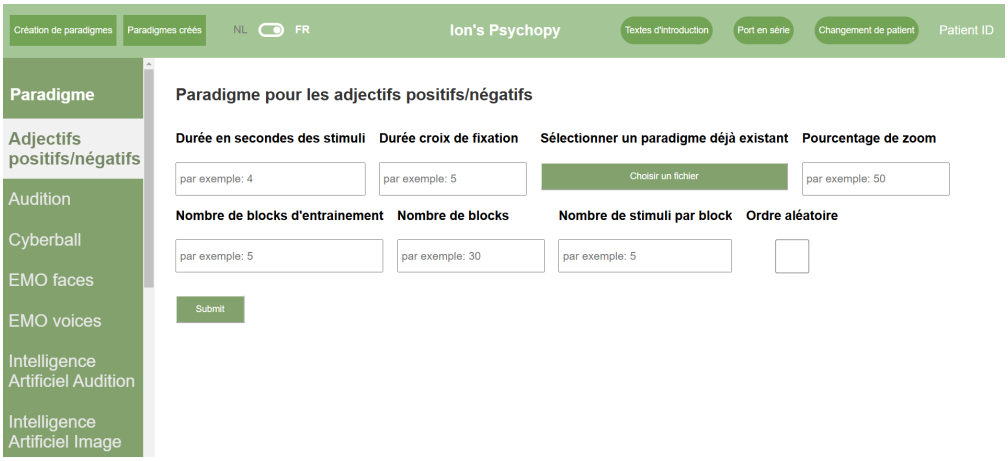

Figure 24: Adjectifs Positive/negative parameters

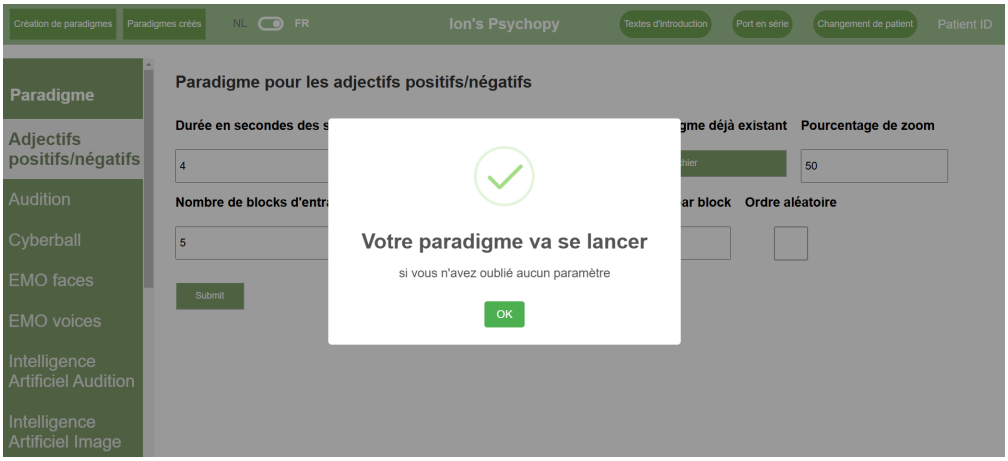

Figure 25: Submitting a paradigm

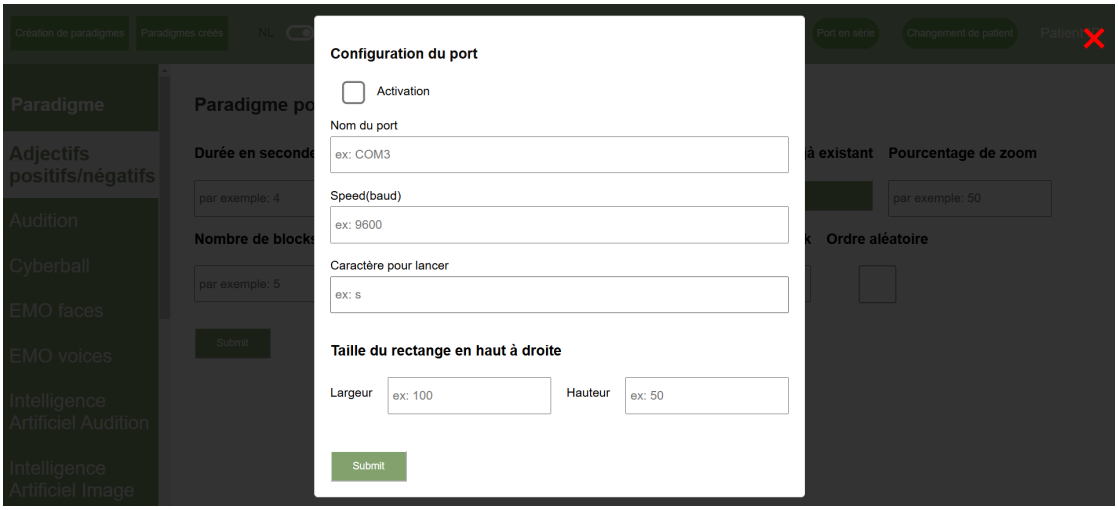

Figure 26: Serial Port overlay

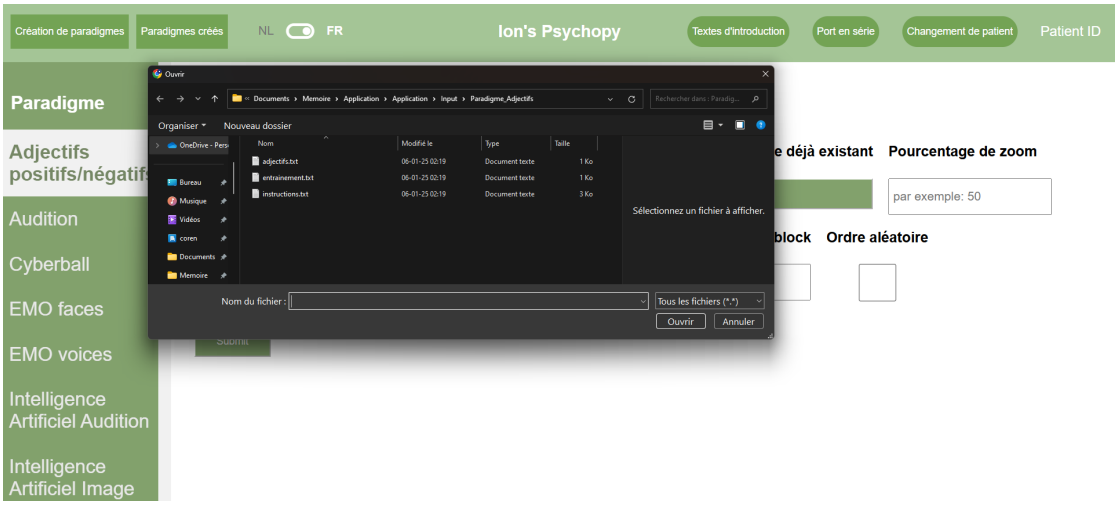

Figure 27: Selecting an introduction slide

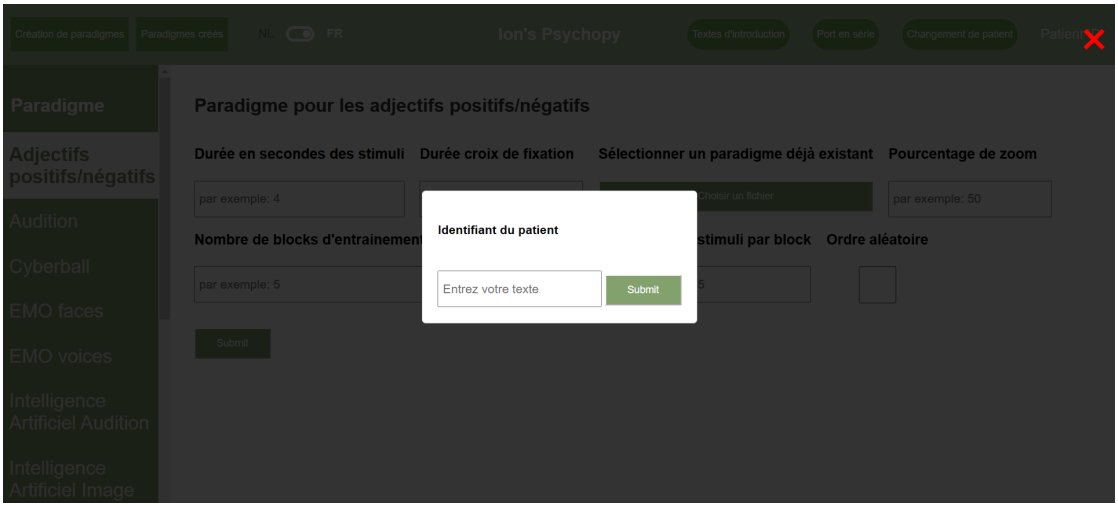

Figure 28: Changing Patient ID

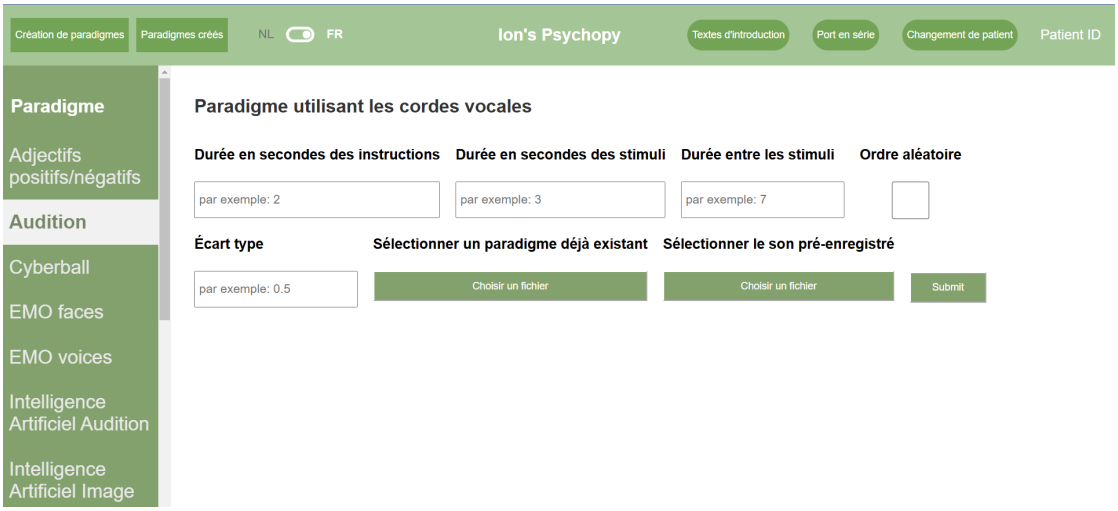

Figure 29: Audition parameters

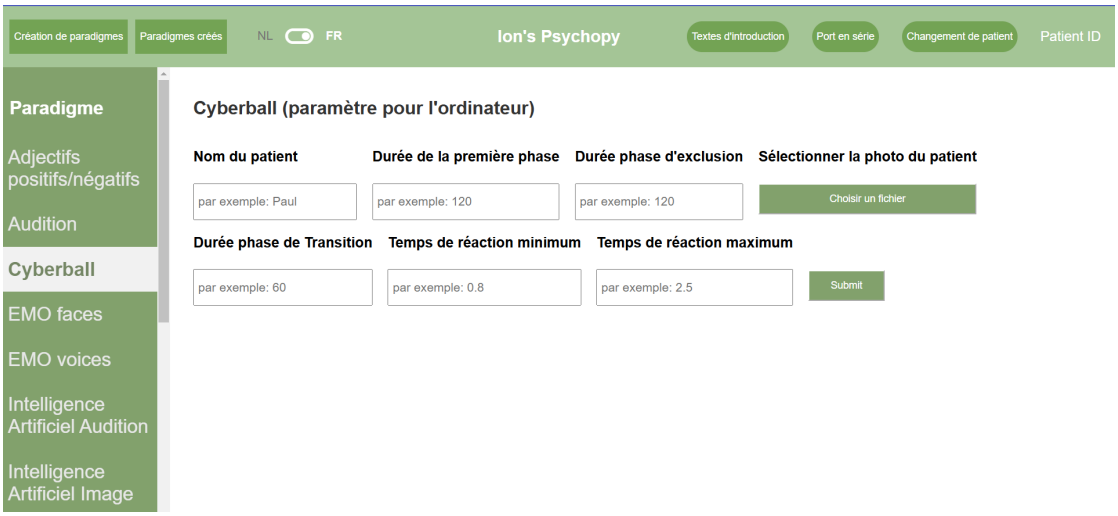

Figure 30: Cyberball parameters

Figure 31: Emo Faces parameters

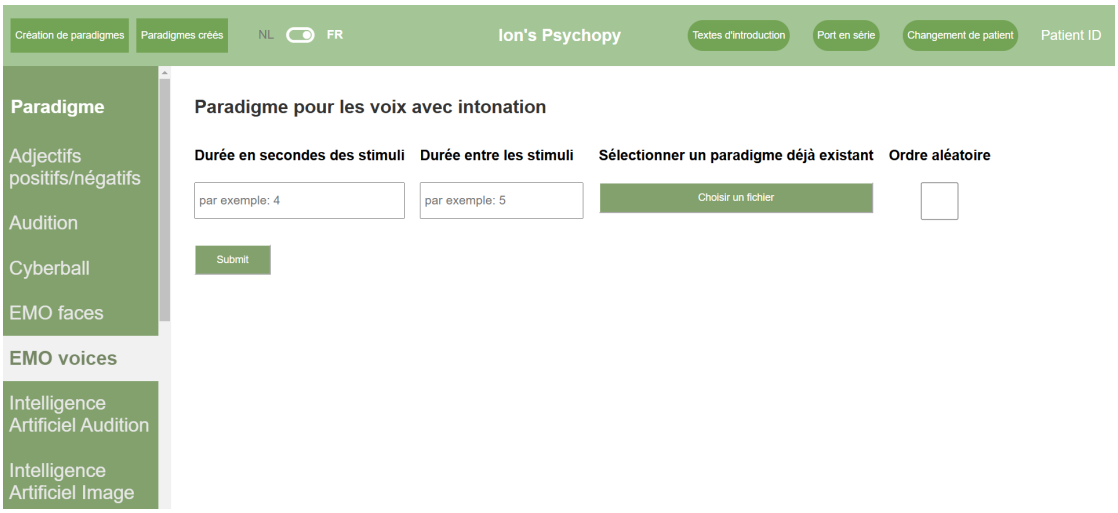

Figure 32: Emo Voices parameters

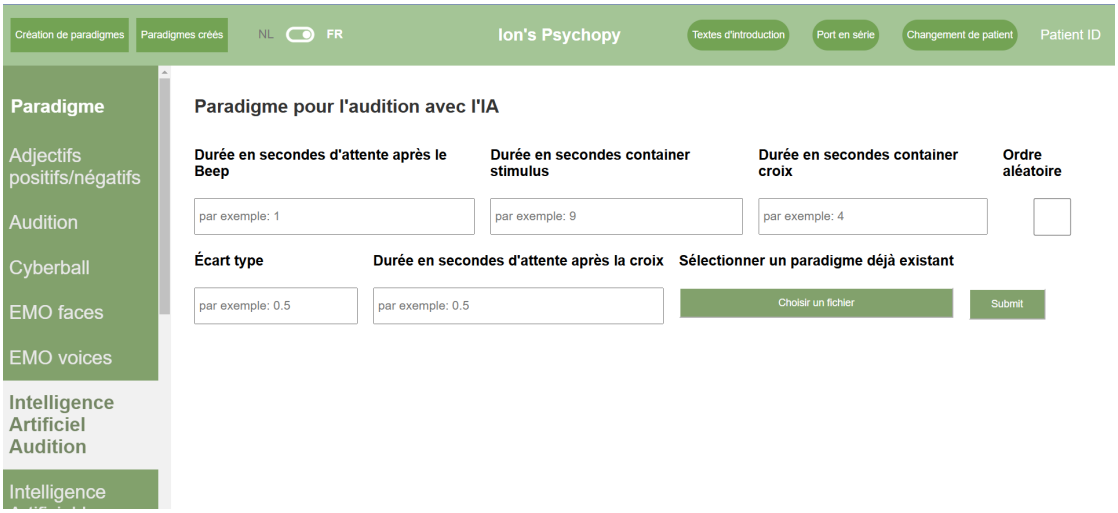

Figure 33: Intelligence Artificiel Audition parameters

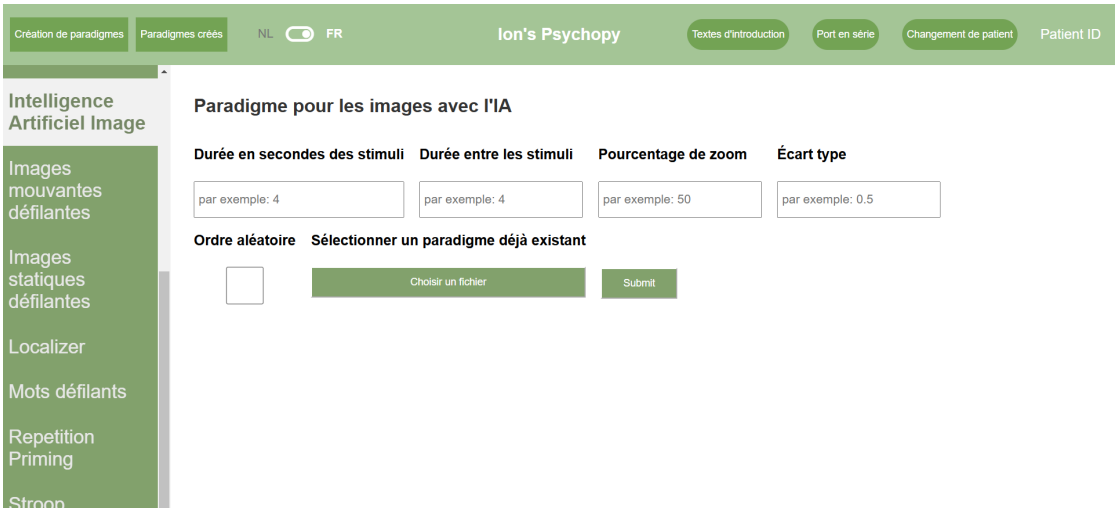

Figure 34: Intelligence Artificiel Image parameters

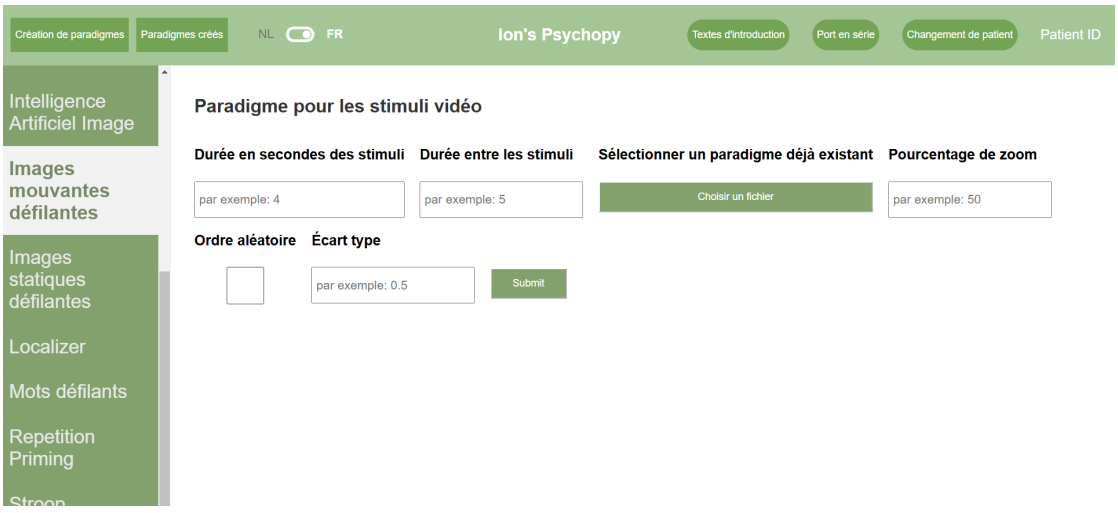

Figure 35: Moving scrolling images parameters

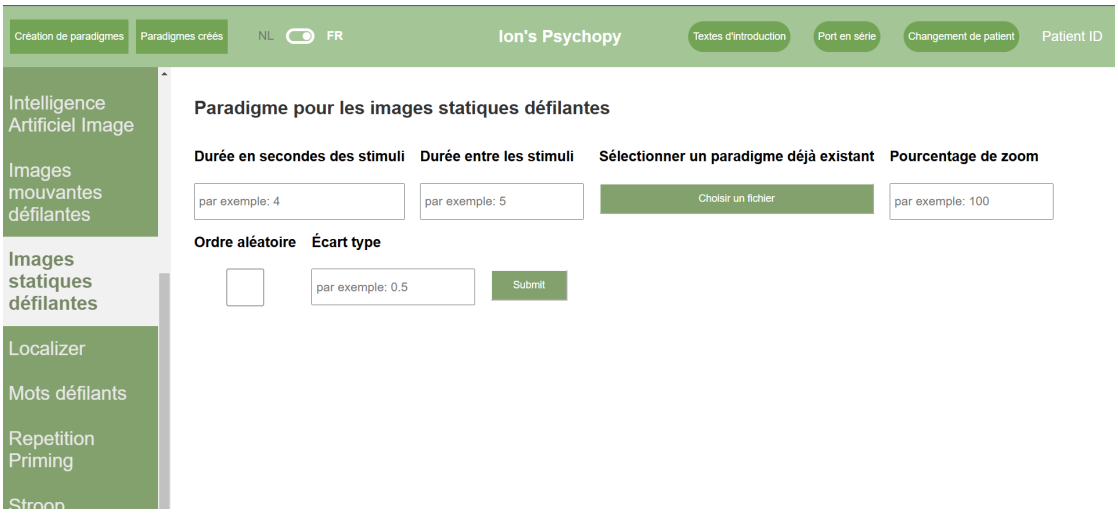

Figure 36: Static images parameters

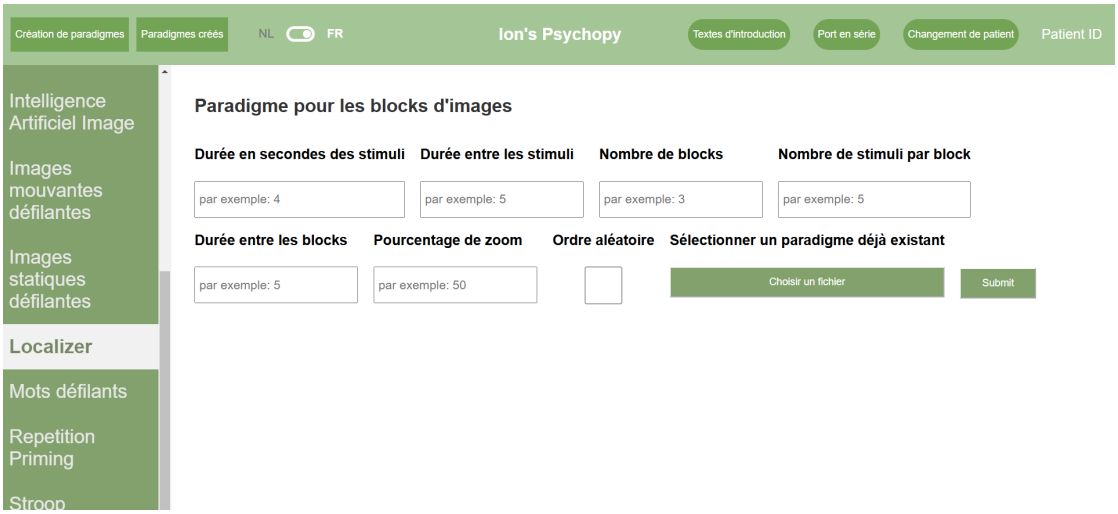

Figure 37: Localizer parameters

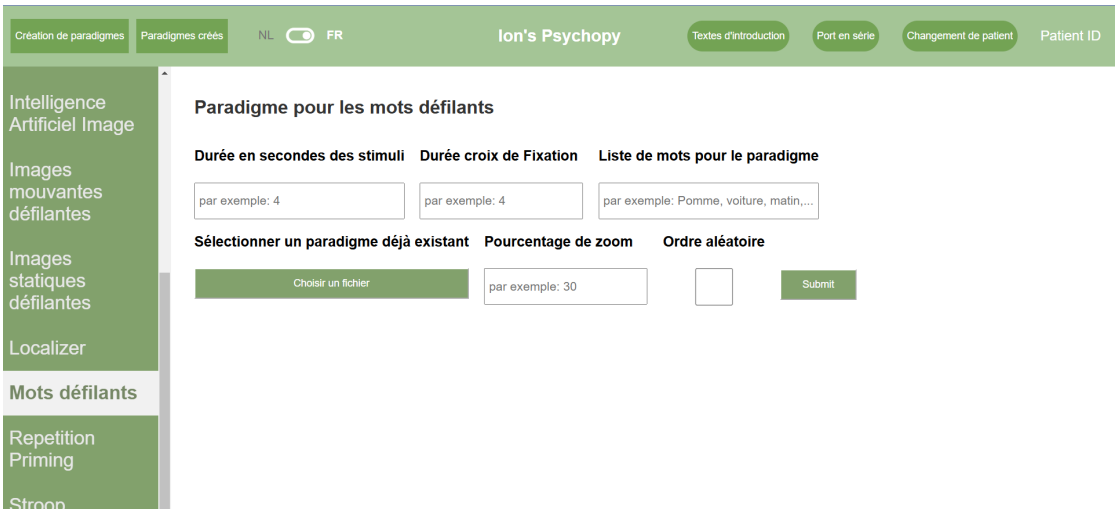

Figure 38: Scrolling words parameters

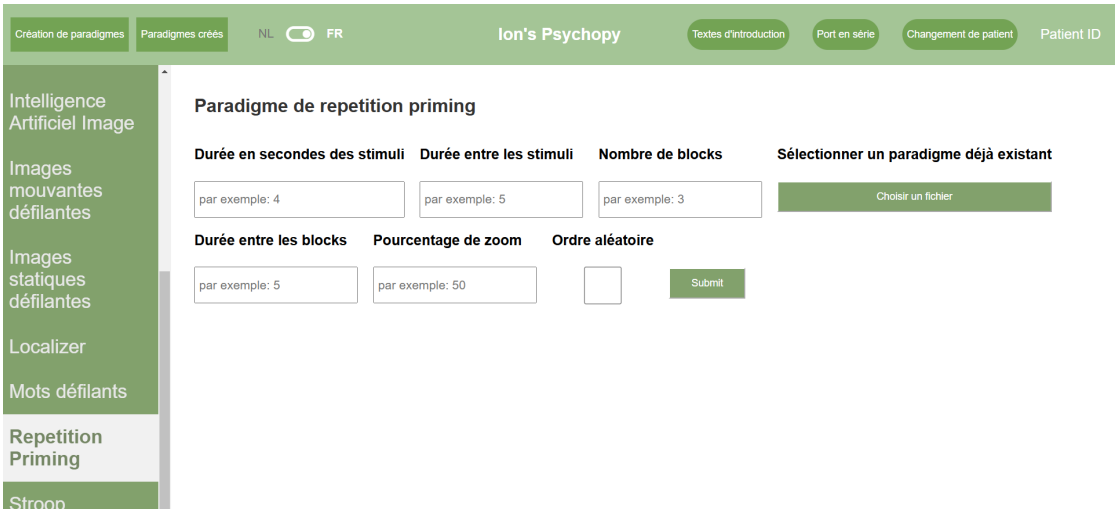

Figure 39: Repetition priming parameters

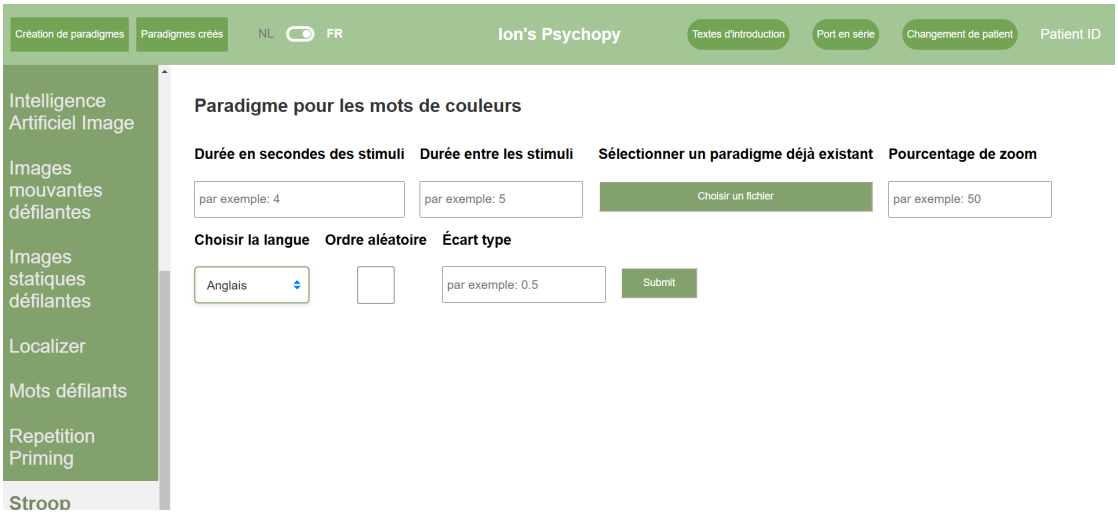

Figure 40: Stroop parameters

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**