Universität Stuttgart

Visualisierungsinstitut

Allmandring 19
70569 Stuttgart

Bachelorarbeit

On-the-go Authoring Visualizations through Wearable Keyboards

Sarah Dosdall

Studiengang:	Medieninformatik
Prüfer:	Prof. Dr. Daniel Weiskopf
Betreuer:	Dr. Leonel Merino, M. Sc. Katrin Angerbauer
begonnen am:	01.04.2021
beendet am:	01.10.2021

Abstract

Nowadays, mobile computing allows users to take advantage of a computer anywhere and anytime. Complex interactions like text entry or programming on-the-go remain challenging. This is due to the lack of input options via keys [FZ94]. One option for solving these problems are chorded keyboards that are connected to the mobile computing device via Bluetooth. The main feature of keyboards is the creation of characters by playing chords and not by single keystrokes. Furthermore, the keyboards have a small size due to their wearability. In this thesis, the chorded keyboard Tap Strap 2 is evaluated as a possible input tool for AVAR, a tool for the visualisation of data in Augmented Reality. For this purpose, a special mapping of the Tap Strap combinations was created that is suitable for programming Pharo code. The focus of the thesis is the design, implementation, and preliminary evaluation of a web application called *Learn2Tap*. This application is designed to help users learn the Tap Strap combinations and train them to write Pharo code using the Tap Strap. The usability of the Tap Strap and *Learn2Tap* was evaluated through a 20-day user study. Users were given 15 minutes daily to use *Learn2Tap*. The outcome of the study was determined through interviews, user tracking, weekly testing and daily saving of learning status. The result of the study is that the users would not choose to use the Tap Strap as an input device because of the difficulties they encountered. However, if they had to learn the combinations of the Tap Strap a second time, they would use *Learn2Tap* again.

Kurzfassung Heutzutage ermöglicht Mobile Computing den Nutzern, überall und jederzeit einen Computer zu nutzen. Komplexe Interaktionen wie die Texteingabe oder das Programmieren unterwegs bleiben jedoch eine Herausforderung. Dies liegt an den fehlenden Eingabemöglichkeiten über Tasten [FZ94]. Eine Möglichkeit, diese Probleme zu lösen, sind Chorded Keyboards, die über Bluetooth mit dem Mobile Computing Gerät verbunden sind. Das Hauptmerkmale dieser Tastaturen sind, das Zeichen durch das parallele drücken mehrere Tasten erzeugt werden und nicht durch einzelne Tastenanschläge. Zudem sind die Keyboards aufgrund ihrer Tragbarkeit handlich. In dieser Arbeit wird das Chorded Keyboard Tap Strap 2 als mögliches Eingabewerkzeug für AVAR, ein Tool zur Visualisierung von Daten in der Augmented Reality, evaluiert. Zu diesem Zweck wurde ein spezielles Mapping der Tap Strap Kombinationen erstellt, das für die Programmierung von Pharo-Code geeignet ist. Der Schwerpunkt der Arbeit liegt auf dem Entwurf, der Implementierung und der vorläufigen Bewertung der Webanwendung *Learn2Tap*. Diese Anwendung soll den Benutzern helfen, die Kombinationen des Tap Strap zu erlernen und sie darin schulen, Pharo-Code mit dem Tap Strap zu schreiben. Die Benutzerfreundlichkeit des Tap Strap und von *Learn2Tap* wurde in einer 20-tägigen Benutzerstudie untersucht. Die Benutzer hatten täglich 15 Minuten Zeit, um *Learn2Tap* zu benutzen. Die Ergebnisse der Studie wurden durch Interviews, user Tracking, wöchentliche Tests und tägliche Speicherung des Lernstatus ermittelt. Das Ergebnis der Studie ist, dass die Benutzer den Tap Strap aufgrund der aufgetretenen Schwierigkeiten nicht als Eingabegerät verwenden würden. Müssten sie jedoch die Kombinationen des Tap Strap ein zweites Mal lernen, würden sie *Learn2Tap* erneut verwenden.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

Nowadays, mobile computing has become an integral part of many people's everyday lives. Mobile computing refers to devices that are portable and have access to wireless networks [FZ94]. The three most important characteristics of mobile computers are portability, communication, and mobility [FZ94]. Devices that fulfil these characteristics include smartphones, laptops, tablets, smartwatches, augmented reality glasses, and virtual reality glasses. Users benefit from these features, as they can use the devices more flexibly and have continuous access to networks. However, mobile computing also poses challenges; many devices that fall into the mobile computing category are 'hand-held devices' and are therefore intended to be portable and small. For this reason, space-consuming buttons and keys are often eliminated in order to maximise screen size [FZ94]. By eliminating the keyboard, an alternative solution for text input must be offered. Possibilities for solving this problem exist in Augmented Reality (AR) and Virtual Reality (VR). Lavalle [Lav15] defined VR as follows: "Inducing targeted behavior in an organism by using artificial sensory stimulation, while the organism has little or no awareness of the interference" [Lav15]. According to Azuma [Azu97], AR can be characterised by three main features, which are "the combination of real and virtual, the interactivity in real time, and the registration in 3-D" [Azu97]. In AR/VR, current solutions include speech recognition, controller input, and gestures. However, these have difficulties such as the user getting tired when using a controller for a long time or that speech recognition is less effective when there is loud background noise [LLY+19]. One possible solution to this problem are chorded keyboards. These are characterised by the fact that they are smaller but still have the full functionality of a fixed keyboard. Chorded keyboards [03a; 03c] have a reduced key set, but these keys can be pressed in combination with several other keys. Depending on which key combination is pressed, a corresponding character is displayed. One possible chorded keyboard which also belongs to the wearable keyboard is the Tap Strap[1], which consists of five rings where each ring is pulled over a different finger of one hand. Depending on the number of fingers touching a surface at the same time, a character is output.

1.1 Problem Definition

A concrete example for the problem of text input is AVAR [MSY+20]. AVAR is an agile situated visualisation toolkit. This toolkit aims to support users through the steps in the data visualisation pipeline. These includes transform data, data mapping and view transformations. The data transformation and visual mapping are implemented using Pharo [2] and Roassal2 [3]. AVAR is used in AR to program data visualisations. A Bluetooth keyboard is used for writing code. This has the

[1]Within the thesis, the name Tap Strap refers to the Tap Strap 2, which is the second generation of the Tap Strap.

[2]https://pharo.org/, accessed 06.09.2021

[3]https://github.com/ObjectProfile/Roassal2, accessed 06.09.2021

advantage over a chorded keyboard that the user works with a familiar input device and does not have to learn the combinations of the chorded keyboard for the different characters. However, the problem with the fixed keyboard is that the user is tied to a surface on which the keyboard is placed. Whereas when using a chorded keyboard, the user can move freely in the room and thereby take full advantage of AR. To help the user use the chorded keyboard, the mapping of the key combinations needs to be adjusted. For example, characters that occur frequently in Pharo and Rossal2 should be given simple key combinations. This was done in the following work for Tap Strap 2. To help the user learn the correct combination of characters, a learning application called *Learn2Tap* was developed and evaluated in this thesis. It is designed to help the user learn the combinations of the Tap Strap. It also prepares the user to write Pharo and Rossal2 code.

1.2 Aim of this Thesis

The aim of this thesis is to adapt the mapping of the Tap Strap to Pharo and Rossal 2 to make it easier for the user to write code in Pharo and Rossal 2. Furthermore, a learning application called *Learn2Tap* is being developed. This is intended to support the user in learning the combination through compact learning units. *Learn2Tap* will also include games in which the user practices writing Rossal 2 and Pharo using the Tap Strap. Finally, the usability of the learning application will be evaluated in a 20-day user study. The Tap Strap was chosen as a chorded keyboard because it has the advantage over other chorded keyboards that text input is done by finger tapping. When a user uses a chorded keyboard with keys, the user in VR has the same problem as with a fixed keyboard: They cannot see which keys their fingers are currently on. Another advantage is that the user does not have to hold an input device in their hand, as the Tap Strap can be pulled over the fingers. This prevents the user from getting tired from holding the keyboard. This leads to the hypothesis:

"To program data visualisations on-the-go, chorded wearable keyboards represent a viable means."

1.3 Structure of this Thesis

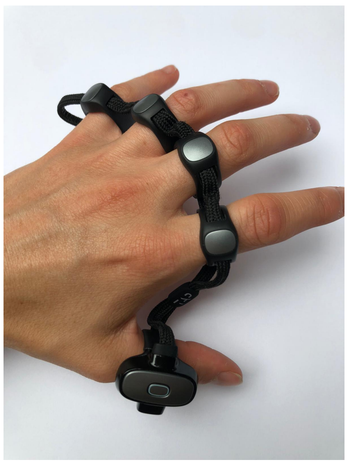
This thesis is structured as follows: In the introduction, I describe the context and main challenges of mobile computing and focus on the problems of user mobility in programming. In chapter 2, I first create the theoretical basis to understand the main concepts that will be developed later in the project. In particular, the chapter provides background knowledge about chorded keyboards, which are the physical keyboards. As well as about virtual keyboards, which represent an option for text input in VR/AR. Chapter 3 describes the technical background and functionality of the Tap Strap and gives a more detailed introduction to AVAR. Chapter 4 presents the methodology of the learning application. This consists of the requirements, design and implementation of *Learn2Tap*. In the section on the design of *Learn2Tap*, the customised mapping for the Tap Strap is also explained in more detail. Chapter 5 evaluates *Learn2Tap* with the help of a user study. The process of the study is outlined, and the results of the study are summarised and critically examined. The thesis ends with a conclusion, a critical appraisal and a discussion of possible outlooks.

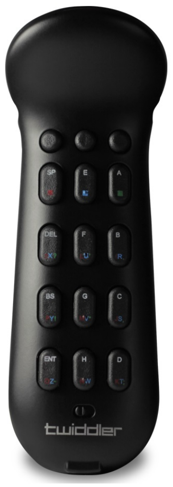
2 Related Work

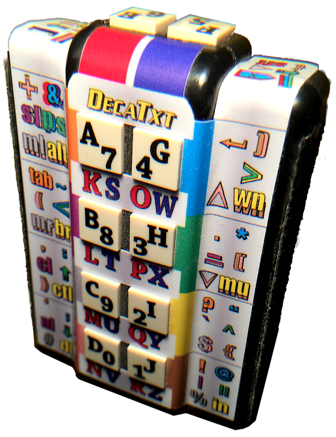
Below, I present and discuss related work to 1) bring the reader up to date with the current state-of-the-art, 2) point out limitations of previous work, and 3) create an understanding of the importance of the research project. The related works are grouped into two topics: ’Physical Keyboards’ [2.1] and ’Virtual Keyboards’ [2.2].

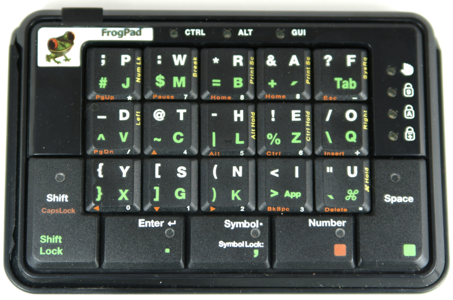
2.1 Physical Keyboards

In addition to the fixed keyboard most used today, there are also wearable keyboards, which are often chorded keyboards. These serve as input devices for computers, smartphones or VR. Chorded keyboards consist of fewer keys than conventional keyboards, e.g. nine keys as shown in Fig. 2.1e.


(a) Tap Strap 2


(b) Twiddler


(c) Decatxt [14b]


(d) FrogPad [03a]

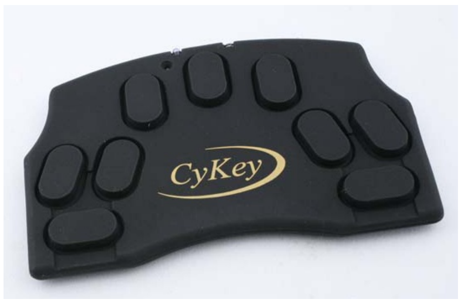

(e) CyKey [14a]

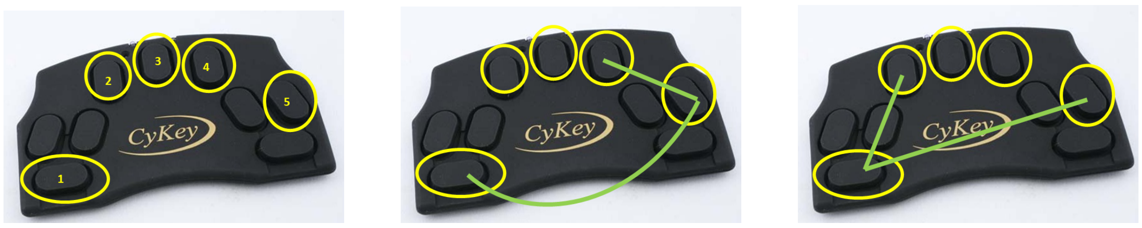
Figure 2.1: This Figure illustrates different types of chorded keyboards.

Different characters can be created by pressing multiple keys simultaneously. For example, if a chorded keyboard has five keys, 5^5 combinations can be created. Since chorded keyboards only have a few keys they are smaller, handier, suitable for one hand and suitable for people with disabilities e.g.: For people with visual impairment [RBS78]. Examples of wearable keyboards are the Tap Strap 2[1], Twiddler[2], FrogPad[3], CyKey[4] and Decatxt[5] (Fig. 2.1). As the number of keys on chorded keyboards is reduced compared to fixed keyboards, characters are created using key combinations, and these key combinations need to be learned. Depending on the manufacturer of the chorded keyboard, different tools for learning the combinations are provided, such as explanatory overview sheets [26a] , video tutorials [03a] , learning apps [26b] or supporting learning software [03b]. This section discusses the learning methods of the Twiddler, FrogPad and CyKey. Section 3.1 explains the Tap Strap and its learning methods in more detail.

The Twiddler[2] (Fig. 2.1b) is a chorded keyboard with 16 keys. These can be customised so that the key combination is tailored to the desired activity. The typing speed of an average user is 30-60 wpm (words per minute) [LSP+04]. The device is designed to fit comfortably in the hand. The combinations can be learned with the help of the "Typing Tutor - Keybr.com[6]". The online tutor designs units that are tailored to the user with the help of algorithms and statistics. The words are based on the phonetic rules of the selected language, making them readable and pronounceable. The length of a unit and the number of letters to be learned can be set in the settings. Another option besides creating exercise units manually, is to let Keybr determine which and how many letters are to be learned. Keybr proceed as follows: First, it collects data on keystrokes, which are used to create statistics. Depending on the statistics, individual letters are learned more or less often. Keybr adds letters individually within a unit. There are fixed thresholds for the typing speed. When this threshold is reached, new letters will be added to the exercise alphabets. The advantages of Keybr over *Learn2Tap* are that artificial intelligence is used to make the practice units variable depending to the user's level. In addition, the user learns to type words directly, which is the aim of learning the combinations. Furthermore, the user can customise the mapping of the Twiddlers key combination in Keybr. *Learn2Tap*, however, has a user-friendly design, an overview of the individual combinations and a playful learning unit. Moreover, before using the combinations, the user receives a tutorial on the combinations and can try them out. By means of games the users can consolidate what they have learned. When visiting Keybr[7], the online typing tutor did not work, possibly because no Twiddler was connected.

Another wearable keyboard is the FrogPad[3] (Fig. 2.1d), which belongs to the category 'one-hand keyboard', the production of this keyboard has been discontinued. The keyboard consists of 20 labelled keys and can be operated with two fingers (thumb and another finger). The user can also set the mapping of the individual keys on this keyboard. "After a few hours of practice, the user should reach a typing speed of 20 wpm" [03a]. For the FrogPad there are manuals, tutorials, and interactive applications. The interactive web tutorials are currently not accessible[8]. After reading

[1]https://www.tapwithus.com/, accessed 26.08.2021
[2] https://twiddler.tekgear.com/, accessed 26.08.2021
[3] http://FrogPad.com/, accessed 26.08.2021
[4]https://sites.google.com/site/cykeybellaire/, accessed 26.08.2021
[5] http://decatxt.com/, accessed 26.08.2021
[6]http://twiddler.tekgear.com/tutor/twiddler.html, accessed 16.09.2021
[7]accessed 26.08.2021
[8]accessed 26.08.2021



(a) Position of the finger. **(b)** Create the letter ’J’. **(c)** Create the letter ’L’.

Figure 2.2: This Figure shows the position of the fingers on the CyKey [14a], as well as the idea of the shape of the letters on the keyboard. The position of the fingers from Figure 2.2a is: 1 = thumb, 2 = index finger, 3 = middle finger, 4 = ring finger, 5 = little finger. For Figure 2.2b and Figure 2.2c, the keys with a green line through them must be pressed to produce the letters ’J’ and ’L’.

the FrogPad user manual [9], the user should understand the structure of the FrogPad. Due to the labeling of the keys, it is not essential that the user knows all key combinations by heart. As there are currently no other learning tools available besides the user manual, the advantages of *Learn2Tap* are automatically given. *Learn2Tap* is an interactive help application for learning the Tap Strap. The user learns the combinations by actively using the Tap Strap. It should be noted that when learning the Tap Strap, the learning application has a higher necessity than with the FrogPad. This is because the combinations are not written down on the device.

The CyKey[4] (Fig. 2.1e) consists of 3 sets of keys, each set contains three keys. Five of the nine keys are used for writing letters. The remaining keys are used for typing symbols and numbers. The typing speed is 40 to 50 wpm and an average user can learn the combinations of the keyboard in 20 minutes [26a]. According to the manufacturer, using the CyKey is intuitive, so there is no special learning tool. As a learning aid, the user gets a graphic [10], showing the alphabet with the corresponding combinations of the CyKey. Furthermore, there is a video [11] which explains how to use the CyKey keyboard. Typing letters works as follows: The user positions the five fingers of one hand on the keys of the keyboard as shown in Fig 2.2a. To type a letter, the keys corresponding to the shape of the letter are pressed. See Figure 2.2 for the letters ’J’ and ’L’. There are no specific learning applications for CyKey. An explanation of the idea behind the key structure of CyKey [12] , an overview of the alphabet and an explanatory video are available. *Learn2Tap* is similar to CyKey’s learning tools in the sense that *Learn2Tap* provides an overview of the tap combination for the symbols. For *Learn2Tap*, there is no explanation of the structure of the combinations, but *Learn2Tap* provides learning units to help the user remember the combinations.

Finally, it should be emphasized that none of the other chorded keyboards presented provides a functioning interactive learning application that supports the user in learning the combinations. In contrast, *Learn2Tap* is a learning application that teaches the user the combinations of the Tap Strap

[9] http://www.FrogPad.com/Images/FP%20PDF%20manuals/Froggy-Manual-USB.pdf, accessed 16.09.2021
[10] https://sites.google.com/site/cykeybellaire/cykey-codecard, accessed 16.09.2021
[11] https://www.youtube.com/watch?v=HBM$_{F}wkMMKE, accessed$16.09.2021
[12] https://sites.google.com/site/cykeybellaire/cykey-home-page, accessed 16.09.2021

in a playful way. The CyKey and FrogPad offer instructions that explain the concept behind the keyboard so that the user only has to understand them. *Learn2Tap* cannot offer such an explanation to the user, as there is no intuitive concept behind the tap combination.

2.2 Virtual Keyboards

Interactions in AR and VR can be performed with the help of controllers, eye tracking, voice recognition and hand tracking. VR gloves are another interaction option in VR. Entertainment through games and 3-D movies are currently the predominant applications of VR [KSF+18]. AR is used, among other things, to project 3-D objects into a room and manipulate them [KPG17]. For AR and VR to establish themselves in other areas such as business and education, there is a lack of good solutions for traditional tasks such as text input that allow users to enter text quickly and accurately [KSF+18]. A previous solution to this problem is text input via a virtual keyboard. A controller is used to select the symbol, and the wpm is 14 [KSF+18]. In comparison, the typing speed with a fixed keyboard averages between 20-40 wpm or 40-60 wpm, depending on the user's ability [HT04]. One problem with typing via controllers is that holding them tires the user [BK19]. This type of text input is available for the Oculus Rift, HTC Vive and Samsung Gear VR [BK19]. With HoloLens, text input can be done using voice recognition, gaze and hand gestures. These input options have several problems [LLY+19]. With hand gesture input, it takes a long time for individual hand gestures to be recognised as corresponding key input. Voice recognition has the disadvantage that it is more difficult to use in a noisy environment and mute people cannot use this type of input. If the input is made via gaze, the user is shown a central keypad in their field of vision. This takes up a large part of the field of vision and thus degrades the AR experience. Due to the problems listed above, the various solutions are not a good alternative to a fixed keyboard. In the following, works are presented that address this problem and present possible solutions in addition to wearable keyboards.

In addition to the existing text entry options with handheld controllers, there are ideas that take this approach and improve it. These advanced controller based text inputs were drum-like keyboard, raycasting, split keyboard and head direction input [BK19]. Here, the user can select the desired symbols on a virtual keyboard using the different input controllers. In a study with 21 participants, it was showen that raycasting and drum-like keyboard are the input tools with the best performance (wpm, accuracy) [BK19]. At the same time, they are also the most preferred input options. The input style of raycasting is "aim and shoot" - virtual rays are moved using two controllers. Once the user has placed the ray on the desired symbol, it can be finally selected with the help of a button. Typing speed was 16.64 wpm and error rate was 11.0%. Study participants enjoyed the shooting technique but had difficulty hitting the correct symbol. In drum-like-key, sticks are used as controllers. The selection of the keys on the virtual keyboard was performed by the down movement of the sticks. This type of text input was first presented by Google Daydream Lab [BK19]. As mentioned earlier, a user study was conducted with the result: The average typing speed was 21.11 wpm, the error rate was 12.11%. The study participants enjoyed the clear feedback on the text input and the playful typing. The active use of both hands was criticised for being tiring. The feedback on the input via controller was positive. Reasons for the positive feedback are the good usability, the positive user experience and the moderate error rate during the first use. A possible extension of the application is auto-completion and auto-correction of text.

Another approach is to enable writing in VR with hand tracking and tracking of the physical keyboard by visualising the hands and keyboard in VR [KSF+18]. Both hands are tracked by self-adhesive retro-reflective markers and subsequently rendered. A Bluetooth keyboard from Apple was used as the keyboard. Four retro-reflective markers were attached to the upper part of the keyboard, allowing both tracking of the keyboard and rendering depending on the physical position. The hands were visualised in different ways, abstract, realistic or as fingertips. A study was conducted with 80 participants, the result of which is that the typing speed of experienced users remains unchanged. The typing speed of inexperienced users decreases by 5-6 wpm. The disadvantage of this type of text input is that it is static. The advantages are that the application is calibration-free, has a low latency and the tracking of the fingers by means of the self-adhesive retro-reflective marker is accurate.

Presenting a virtual keyboard in conjunction with hand tracking is a more advanced research approach. In addition, the position of the virtual keyboard is variable, it can be displayed in mid-air or on a physical surface. Another variable is the number of fingers used for typing. A study with all these different variables was conducted [DBWK19]. The typing style was divided into typing with 10 fingers and typing with 2 fingers (index fingers of the right and left hand). The hand movements were tracked using an OptiTrack[13] motion capture system. The virtual keyboard is based on the QWERTY layout with a reduced character set. Spherical collision bodies attached to the fingertips were used to track whether the virtual keyboard was touched. When a touch event occurred, the user received a click sound to confirm the interaction. Further, an auto-correction feature was integrated into the writing program. A user study was conducted with 23 participants, with the result that placing the virtual keyboard on a physical surface achieved better results. Many users had problems using all 10 fingers when positioning the keyboard in mid-air. This kind of text input is a mobile and easy way to write within VR, but the currently available technologies of remote finger tracking and head mounted tracking are still too inaccurate [DBWK19].

The virtual keyboard approaches do not require a learning application, as no special key combinations need to be learned. The handheld controller methods require the user to learn how to use the controller. The idea has the disadvantage that users find the use of the controller tiring [BK19]. The hand tracking approaches address the problem that the current hand tracking available on the market is still too inaccurate [DBWK19]. Furthermore, one of the approaches uses a physical keyboard, which takes away the users mobility. A similar problem arises with the virtual keyboard, as users found the use of a virtual keyboard mid-air less comfortable than a virtual keyboard projected onto a surface [DBWK19]. The advantage of using a Tap Strap is that the user is not tied to a surface.In addition, the Tap Strap 2 is already available on the market and can be used in VR and AR.

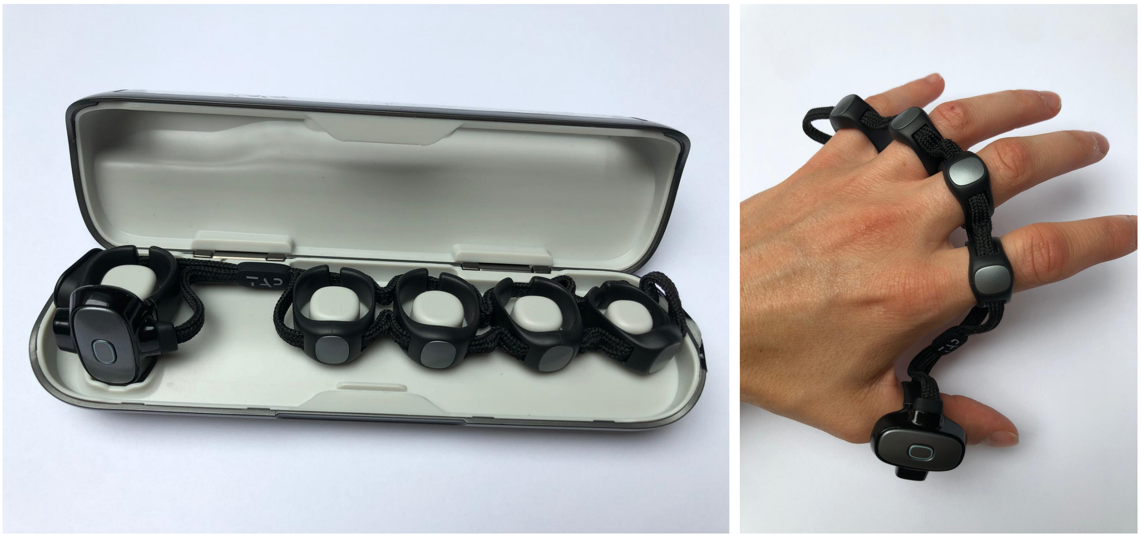
[13], https://www.optitrack.com/, accessed 21.09.2021

3 Background

Continuing the thematic introduction, the theoretical basics will now provide the necessary knowledge so that the reader can fully comprehend the course of the elaboration. First, Tap Strap 2 will be discussed. Then AVAR will be explained.

3.1 Tap Strap 2

The following section the wearable keyboard Tap Strap 2 (see Fig. 3.1), which is the subject of the work, is explained in more detail. The device has 3 functions: Air Gestures, Optical Mouse and Text Input by tapping, and can be connected to a device via Bluetooth. The tap combinations too tapping letters and symbols can be fully personalised with the Tap Mapper tool [1]. An average user has a typing speed of 35 wpm [2]. The combinations can be learned with the help of the apps



(a) Tap Strap 2 in charging box. **(b)** Tap Strap 2 on hand.

Figure 3.1: The Figure on the left shows the Tap Strap in its charging box. The right Figure shows how the Tap Strap is worn. The hand position corresponds to the position of the hand when typing with the Tap Strap.

[1]https://map.tapwithus.com/, accessed 26.08.2021

[2]https://www.tapwithus.com/how-tap-works/, accessed 26.08.2021

Right Combination	Left Combination	Symbol
● ○ ○ ○ ○	○ ○ ○ ○ ●	A
○ ● ○ ● ○	○ ● ○ ● ○	M
○ ● ● ● ●	● ● ● ● ○	H

Table 3.1: Example showing the representation of the combination with the corresponding symbol.

Tap Manager[3], Tap Genius [4] and Tap Academy [5]. Tap Manager is used for technical management and basic introduction of the Tap Strap. With the help of Tap Genius, the user learns the default combinations of the Tap Strap. The Tap Academy application offers the user a 10-minute practice session on the combinations learned from Tap Genius for 30 days. In the following, the technical background and different learning applications will be discussed in more detail.

The Tap Strap 2 keyboard consists of five flexible rings. Each ring contains a three-axis accelerometer to register the movement of the fingers. The individual rings are connected by a braided nylon strap, which includes conductors. This ribbon is used to transmit the signals from the individual accelerometers to the circuit board. The circuit board is installed in the thumb ring and a microcontroller collects incoming data. This microcontroller converts the data into combinations, which are then sent to the connected device via Bluetooth. Furthermore, the thumb ring contains haptic elements, a six-axis inertial measurement unit, a battery and an optical mouse.

The Tap Strap 2 has a total of three modes: Tapping Mode, Optical Mouse Mode and Air Gesture Mode. These modes are explained in the sections below.

Tapping. The Tapping Mode, which is the default mode of the Tap Strap 2, allows the user to type letters and symbols. Tapping the combinations has similarities to playing a piano. For example the letter 'A' is created by a Single-Tap of the thumb (see table 3.1). The letter 'M' is tapped by simultaneously tapping the index finger and ring finger, and the letter 'H' is written by tapping the index finger, middle finger, ring finger and little finger in parallel.

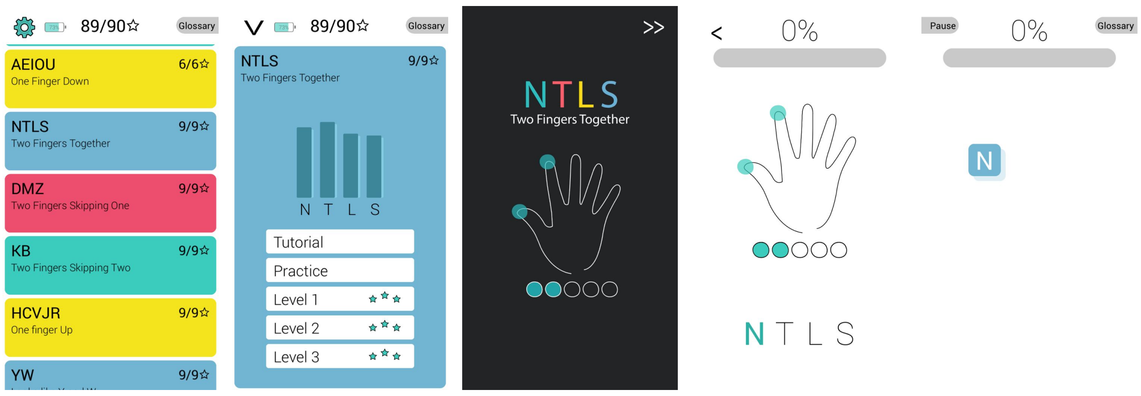
The tapping mode can be divided into five tap modes, the Single-Tap, Double-Tap, Triple-Tap, Shift and Switch mode. This results in a total of 155 combinations, some of the combinations have fixed functions that cannot be changed. With the help of Tap Mapper, most of the combinations in the different tap modes can be personalized. Besides letters and symbols, combinations can also be keyboard shortcuts, special symbols and modifiers.

Optical Mouse. Another mode of the Tap Strap is the Optical Mouse. This mode is activated when the thumb is placed on a surface. Small movements of the thumb move the cursor. Mouse clicks are performed by tapping the fingers. Tapping the middle finger simulates a 'Right Mouse' click. Tapping the index finger represents the 'Left click'. All basic mouse commands can be simulated in this mode by the corresponding finger taps, including scrolling and drag-and-drop. To return to tapping mode, the hand must be lifted from the surface.

[3]https://apps.apple.com/us/app/tapmanager/id1225883603, accessed 26.08.2021

[4]https://apps.apple.com/us/app/tapgenius/id1103667402, accessed 26.08.2021

[5]https://apps.apple.com/de/app/tapacademy/id1424336981, accessed 26.08.2021



(a) Main menu **(b)** Learning Unit **(c)** Tutorial **(d)** Practice **(e)** Level 1

Figure 3.2: The Figures above (a - e) are an excerpt from the app Tap Genius[4].

Air Gesture. The last mode is the Air Gestures Mode. To switch to this mode, the hand must be positioned as if the user wanted to give someone a hand shake. This mode supports the functions of a standard mouse. By moving the hand, the mouse cursor is moved. Extending one finger and swiping to the left or right will perform mouse clicks. The user can scroll by extending two fingers. The direction of scrolling is determined by the direction of movement of the extended fingers.

The localisation of the fingers, as well as the orientation and movement of the hand in the Air Gesture Mode, is supported by the 6-axis IMU integrated in the thumb and the 3-axis accelerometer in each ring.

3.1.1 Learning Applications

The manufacturers of the Tap Strap offer three apps to help users learn how to use the Tap Strap. The Tap Manager is mainly used to configure the device. Tap Genius teaches the user the standard letters and symbols. Once the users have completed all the learning units of Tap Genius, they can consolidate the learned knowledge with the help of Tap Academy.

Tap Manager[3]. The first app introduced is the Tap Manager, which allows you to manage the settings of the Tap Strap. For example, whether the user is left- or right-handed. There is also a short introduction (which cannot be skipped) on how to use the Tap Strap. The app is an important part for the *Learn2Tap* application, because here you can configure which mapping the Tap Strap should uses. Specifically, this means that within the app, the customised mapping for the *Learn2Tap* application must be downloaded and then selected as the mapping to be used. If the app is used under iOS, no problems occur. When using the app from the Play Store, it is not possible to select a personalised mapping because the app reports that the Tap Strap needs an update, but also reports that there are no updates. This problem occurred in both study participants using an Android device[6].

[6]as of July 2021

Tap Genius[4]. The Tap Genius app (see Fig. 3.2) is designed to enable users to learn the entire tap alphabet in less than two hours. There are a total of eight learning units in which the various combinations are learned. A learning unit (Fig. 3.2b) lasts about 10 minutes and contains a Tutorial Part (Fig. 3.2c), Practice Part (Fig. 3.2d) and Level 1-3 (1-2) (Fig. 3.2e). In the learning units, punctures are introduced within the levels in addition to the letters. There is also a practice section for Air Gesture and Mouse Mode, as well as a Speed Challenge and an endless Mode. Tap Genius is designed to teach the user the default mapping, which means that personalised mappings cannot be learned with the app.

Tap Academy[5]. The last app that will be presented is Tap Academy, which is designed to help users consolidate the knowledge they have learned from the Tap Genius app. The duration of Tap Academy is 30 days and users are encouraged to play 10 minutes of games/challenges every day to consolidate their knowledge. The users get feedback about their completed learning unit by means of achieved points, which also considers the accuracy. The app includes a statistics part, where the user can see the achieved points of the games per day. In addition to the prescribed learning units on a day, the user has the possibility to play other games in the Practice Part.

Difference to *Learn2Tap* The main purpose of Tap Manager is device management. Thus, the app has different functionalities than *Learn2Tap*, as the aim of *Learn2Tap* is to teach the user the combinations. *Learn2Tap* does not provide device management options. When comparing *Learn2Tap* with Tap Genius, it is noticeable that the learning units have the same structure. The reason for this is that the structure of the learning units of *Learn2Tap* is based on that of Tap Genius, but *Learn2Tap* has additional functions. *Learn2Tap* teaches the user all 5 tap modes, while Tap Genius is limited to a total of 8 learning units, which mainly teach the Single-Tap mode. Since the mapping of the combinations is stored in the *Learn2Tap* database, *Learn2Tap* can teach the user different mappings. To do this, the contents of the tables *levels* (see Fig. 4.15) and *overview* (see Fig. 4.16) have to be adapted. Another advantage of *Learn2Tap* is that it is a web application, so it does not have to be installed locally on a device. The user has the possibility to use the web application on any device with the same learning status. The game part of *Learn2Tap* is based on two games from Tap Academy. The Tap Academy game 'Streak Challenge' is similar to the *Learn2Tap* game *Time To Tap*. The difference is that 'Streak Challenge' only practices the symbols that the user has learned in Tap Genius. The *Time To Tap* game gives the user the opportunity to practise the combinations of the different modes individually or in combination. The game *Tap it 5-Times* from *Learn2Tap* is based on the game 'Repeat Five' by Tap Academy. The difference is that the words in *Tap it 5-Times* are based on words that occur in programming with Pharo. *Learn2Tap*'s last game *Coding* is not based on any of the Tap Academy games. As this game is designed to prepare the user for programming in Pharo, which Tap Academy does not do. Furthermore, symbols or sentences to be practised in *Learn2Tap* can be adapted in the database for different mappings.

Tap Mapper[1]. The Tap Mapper allows you to create personalised mappings or download mappings that other users have created. These mappings are then adapted for special purposes, such as for the game Fortnite, to operate Spotify, Netflix or for programming in C. The mapping of the combinations depends on the HID code, which indicates the position of a key on the keyboard layout is. This means that a mapping that is designed for a German keyboard layout is not compatible with the US keyboard layout. For example, the computer is set to the German keyboard layout and the Single-Tap of the thumb results in a 'z' if the keyboard of the computer is changed to the US keyboard layout, the Single-Tap of the thumb now results in a 'y'. This is especially important when mapping symbols, since the layout of the German and US keyboards is very different.

3.2 AVAR

This section explains the Agile Situated Visualization toolkit AVAR [MSY+20], which is used in AR. The toolkit allows the user to perform "data transformation, visual mapping and view transformation"[MSY+20] within the immersive environment.

A special focus in the development of the toolkit was to get live feedback while creating and modifying visualisations. Furthermore, the user should be able to perform all visualisation processes without leaving the integrated environment. The user can process data in 3 steps ("Data Transformation, Visual Mapping, View Transformation"[MSY+20]) using the toolkit. The Data Transformation is the first step of processing. Here the user has the possibility to transform a given raw dataset using different functions and to create tables with this data. The transformations are based on Pharo[7]. The next step involves visual mapping. The data tables are transformed into a visual structure. Wooden [8] and Roassal2[9] are the builders used for this. The last step is the transformation of the view. The rendered data can be viewed within the AR. Moreover, the view can be transformed via a user interface or program-controlled. The view can be rotated by hand gesture. To relocate the view, hand gesture can be used in combination with walking.

Interaction with the toolkit can be performed using the Bluetooth keyboard, body movements, hand gestures with airtap and head movements. The Bluetooth keyboard is used to interact with the sample browser, the code editor and the console panel. Another option offered by the keyboard, is scrolling through the code. Besides the keyboard, hand gestures can also be used to scroll through the code. When hand gestures are used together with an airtap, the user can rotate the visualisation. Hand gestures, air tap, head and body movement can be used to reposition the visualisation. To get contextual information, the user can hover over elements by moving their head.

A possible approach for interaction with the toolkit would be the Tap Strap, as it allows the user to use air gestures, to move freely in the room and also allows complex interactions such as programming with the device. To be able to write code fluently with the Tap Strap, users must first learn the combinations of the Tap Strap. For this purpose, *Learn2Tap* was developed to help users learn the combinations of the Tap Strap. In this case, the combinations of the mapping that support programming in AVAR are learned. This means characters that are often used in AVAR occupy a simple combination. Furthermore, there are games in *Learn2Tap* that prepare the user to write code using Tap Strap. Details about *Learn2Tap* can be found in the following section.

[7]https://pharo.org/, accessed 06.09.2021
[8]https://github.com/ronsaldo/woden, accessed 10.09.2021
[9]https://github.com/ObjectProfile/Roassal2, accessed 06.09.2021

4 Learn2Tap

This chapter introduces the *Learn2Tap* web application which supports the users in learning the combinations of the Tap Strap and prepare them to write Pharo code using the Tap Strap. For this purpose, the mapping of the Tap Strap was adapted. First the requirements are defined, then the design of the learning application and the design of the tap mapping are presented. Finally, the implementation will be discussed.

4.1 Requirements

This section explains the framework of the learning application *Learn2Tap* and to includes all necessary requirements. For this purpose, all necessary functions are discussed first. A general distinction is made between functional and non-functional requirements [HB13]. The functional requirements are those that are necessary for the learning application to function. They determine, what the system is to do from a technical point of view.

The non-functional requirements, in contrast, define the technical behaviour of a system. Furthermore, they describe the quality in which the learning application should perform and are, for example, requirements for the reliability and usability of the application. Section 4.1.3 contains the introduction of the user group and thus completes the chapter Requirements.

4.1.1 Functional requirements

The recorded requirements are described below using natural language. The following functional requirements were identified:

FR.01: Global Availability. *Learn2Tap* should be accessible from different end devices at any time. Must have the same user status on all end devices.

FR.02: Create/ select users. In order to be able to assign the learning progress of different users, there must be the possibility to create a user and to be able to select this user when logging in from another end device.

FR.03: Saving the learning progress of individual users. The learning application must be able to store and assign the learning progress of different users.

FR.04: Suitable for right-handed and left-handed user. *Learn2Tap* offers the user the possibility to choose between right-handed and left-handed. Depending on the selection, the display of the combination must be adapted.

FR.05: Intuitive user interface. In order to make the user interface intuitive and comprehensible the 8 Golden Rules of Interface Design by Shneiderman [Shn98] apply. These rules are:

- “Strive for consistency.
- Enable frequent users to use shortcuts.
- Offer informative feedback.
- Design dialogs to yield closure.
- Offer simple error handling.
- Permit easy reversal of actions.
- Support the internal locus of control.
- Reduce short-term memory load.” [Shn98]

FR.06: Overview of Tap Strap combinations. There must be an overview of the mapping of the Tap Strap.

FR.07: Learning the Tap Strap combinations. The user learns the Tap Strap combinations through small *Learning Units*, which can be repeated as often as desired. A *Learning Unit* must teach 2-5 new characters, and include a *Tutorial*, *Practice* and *Level Part*.

FR.08: Design of the *Tutorials*. In the *Tutorial*, the user is shown which combination results in which character.

FR. 09: Design of the *Practice Part*. In the *Practice Part*, the user practices the new characters.

FR. 10: Design of the *Levels*. The *Level Part* must consolidate the newly learned characters. For this purpose, the next character is chosen randomly. Characters from previous *Learning Units* can also be used. A *Learning Unit* contains 2-3 *Levels*, with increasing difficulty.

FR.11: Consolidate the learned knowledge. *Games* must consolidate the learned knowledge and prepare the user for programming with Tap Strap in Pharo.

FR.12: Live feedback. The user must receive feedback when the tapped combination is correct. For example, the character to be typed should flash red if the input is incorrect.

FR.13: User gets feedback about what has been learned. At the end of a *Game* or *Level* the user must receive feedback on the completed *Learning Unit*.

FR.14: Logging of user interactions. To perform the user study, the program must document all user interactions, i.e. which button or key was pressed and how much time it took the user to tap a certain character.

FR.15: Preparation for programming in Pharo. The user should be able to program with the Tap Strap in Pharo after completing all *Learning Units* and playing the *Games*. For this purpose, a game must be created that teaches the user how to program in Pharo using the Tap Strap.

4.1.2 Non-functional requirements

While functional requirements are established on a project-specific basis, non-functional requirements are grouped and specified by ISO/IEC 9126 "Quality Characteristics for Software". This standard includes the following application requirements: Maintainability, Usability, Efficiency, Functionality, Transferability and Reliability of the system. This must be implemented in the learning application, in addition to the functional requirements, to ensure the quality of the application.

4.1.3 User group

The user group is divided into two groups of people: the developer and the end user. The user interface looks the same for both user groups. The developer modifies the web application manually via the code. The administration of the application is done on the computer/server where the application is installed. The end user can create and select a user. Once the user is logged into *Learn2Tap* with a user, they have the option to learn new characters, play games, as well as view the statistics of what they have learned so far. Furthermore, another user can be selected at any time.

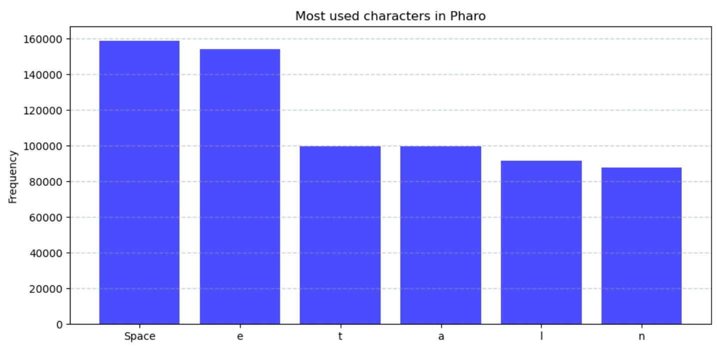
4.2 Design

For the implementation of the learning application a web application was chosen, which is built according to the client-server model. This has the advantage that the application is platform-independent. In addition, the users do not have to install the application locally on their computer before using it but can access the application via a web browser. Furthermore, changes to the program only need to be made on the server on which the application is installed. The data of the users were stored in a database schema. The connection and modification of the databases was realised in the server. The following sections describe the design of the Tap Strap Mapping, client, server and database in more detail.

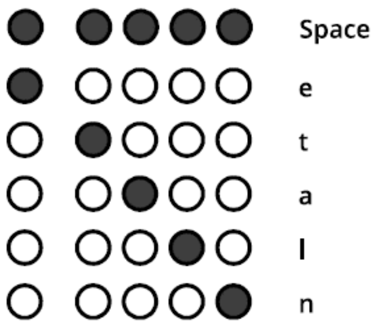
4.2.1 Custom mapping

The purpose of this section is to explain the basis on which the Tap Strap mapping was designed. As well as what special features in Double- and Triple-Tap mode need to be taken into account when creating the web application.

Tailoring the Mapping. The mapping which was created for the work is used for programming in AVAR (Section 3.2). In AVAR, programming is done in the Pharo language. In dependence of a statistic (appendix A.1), which shows how often which characters are used in AVAR, the mapping was adapted. The diagram 4.1a shows the six most frequently occurring characters within Pharo, which then received a well executable combination within the mapping (Fig. 4.1b). The more frequently a combination is executed the easier it is to tap the character. As explained in section 3.1 the Tap Strap has five tapping modes. In the mapping created for AVAR, the modes were divided as follows: The Single-Tap mode allows the alphabet to be written. In the Double-Tap mode, capitalisation can be activated, there are easier combinations for letters that had a difficult combination in the Single-Tap mode (e.g. the 'w') and symbols can be tapped. The Triple-Tap mode



(a) Diagram showing the most used characters in Pharo.



(b) Tap combinations and their character (view for right handers).

Figure 4.1: The frequency of the most used characters from Pharo is shown on the left. The combinations of the most frequently used characters are shown on the right. These have a particularly simple tap combination.

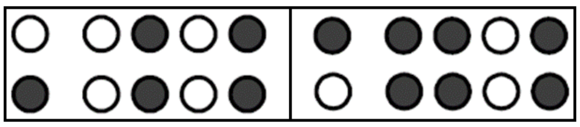


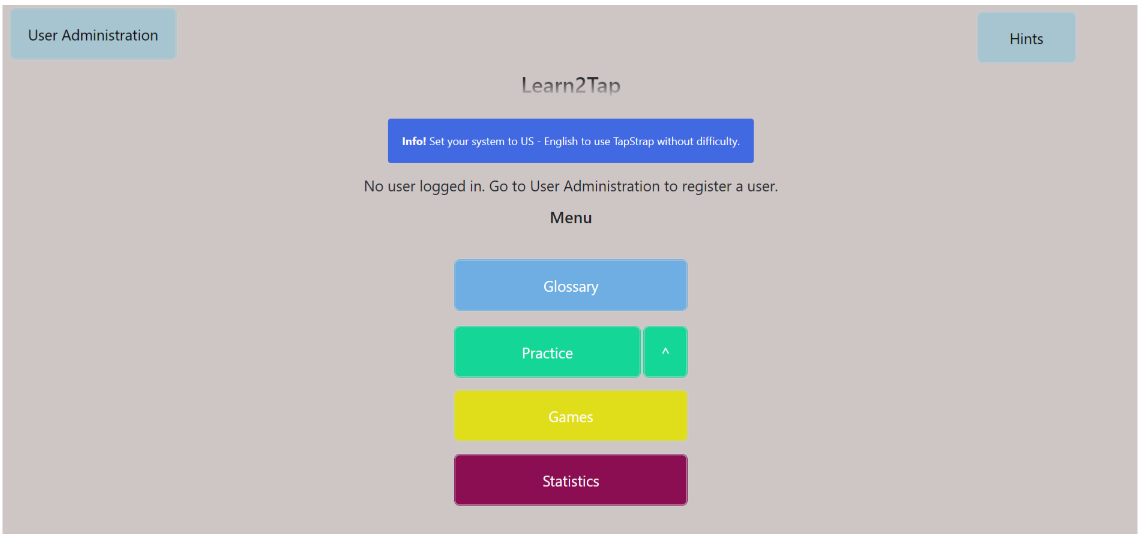
Figure 4.2: Examples of difficult combinations (view for right handers).

is also used to tap symbols. Shift mode contains useful shortcuts for the AVAR environment such as 'Find Class' or 'Create test method and jump to it'. The Switch mode includes the numbers, arrow symbols, as well as the ability to select, copy and paste text. In all modes except the Single-Tap mode, only the combinations that are easy to tap are used. Examples of difficult combinations are shown in fig. 4.2. The whole mapping can be viewed in the appendix A.2.

Double-Tap and Triple-Tap mode. The Double- and Triple-Tap modes are executed by the user tapping the fingers for the corresponding combination twice or three times in quick succession. As a result, when the user performs a Double-Tap, the Single-Tap character is displayed first, which is then cleared using Backspace, and then the Double-Tap symbol is displayed. The same happens in the Triple-Tap. For example, when tapping with the little finger the output is 'n', when performing a Double-Tap the end output is ' / ', but the total output is ' n Backspace / '. If a Triple-Tap is performed with the little finger, the last character displayed is a '∧', the total output performed during the tap is 'n Backspace / Backspace Shift ∧'. The total output must be taken into account when implementing *Learn2Tap*, otherwise the error rate and accuracy in Double- and Triple-Tap mode will not be correct.

4.2.2 Client Application

The client represents the front-end of *Learn2Tap* and is thus the interface of the application. The application can be accessed by entering a URL in any web browser. Entering tap combinations and button clicks, the user can interact with the application. Fig. 4.3 shows the 'Welcome Page'. Depending on whether a user is logged in, the 'Welcome Message' changes. As can be seen from

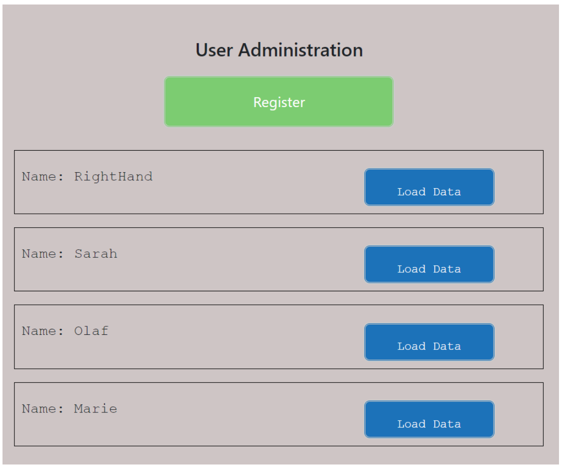


Info! Set your system to US - English to use TapStrap without difficulty.

No user logged in. Go to User Administration to register a user.

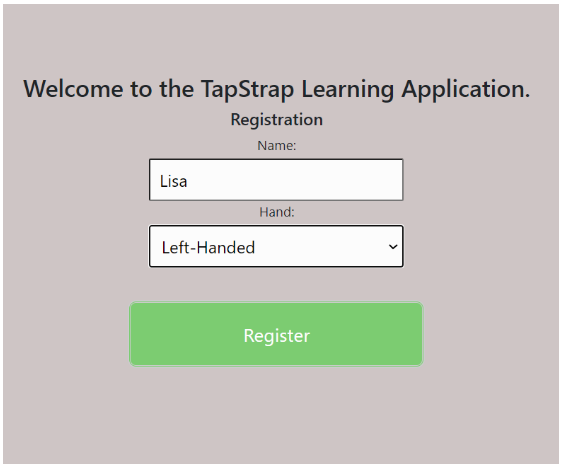
Figure 4.3: The *Learn2Tap* main menu is displayed here. This is the view when no user is logged in.

Fig. 4.3 *Learn2Tap* is divided into four *Main Menu Items* (*Glossary*, *Practice*, *Games*, *Statistic*) and two *Side Menu Items* (*User Administration*, *Hints*), which will be explained in more detail below, starting with the *Side Menu Items*.



Name: RightHand
Name: Sarah
Name: Olaf
Name: Marie

(a) *User Administration.*

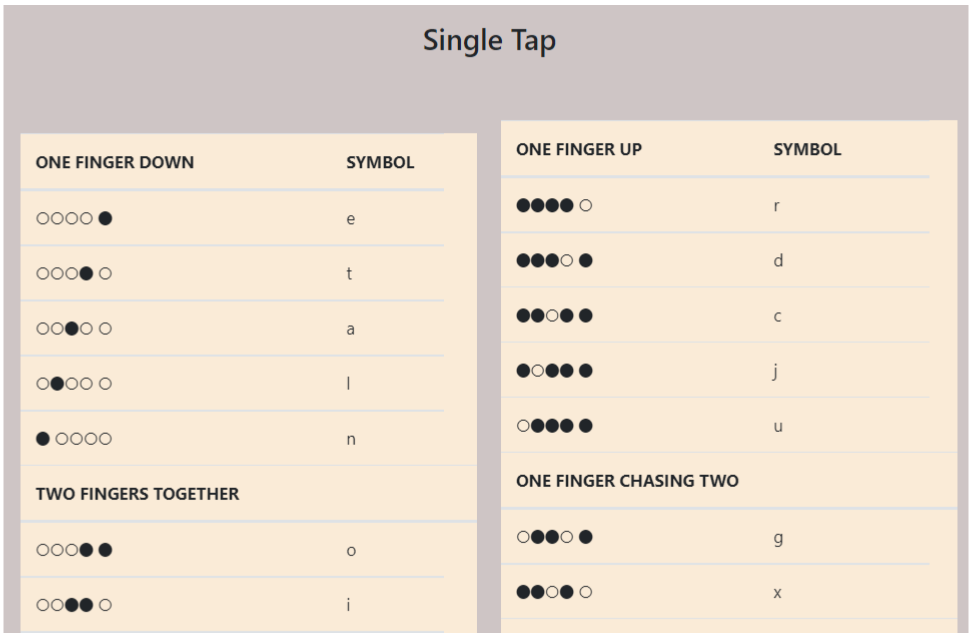

Welcome to the TapStrap Learning Application.

Name: Lisa
Hand: Left-Handed

(b) User Registration.

Figure 4.4: Figures (a) and (b) are part of the *User Administration Side Menu Items*. Here you can select (a) or create (b) a user.

User Administration. First, the *Side Menu Item User Administration* is explained. Within this menu option, a user can be selected or newly created (Fig. 4.4). If an already existing user is selected, *Learn2Tap* loads the data of the corresponding user from the database into the application. This data contains the learning status of the user, as well as which hand is being used to type. When creating a new user (Fig. 4.4b), a name must be selected that does not yet exist as well as the hand with which the combinations are practised. In case a username is selected that already exists, a dialog will open informing the user that the username already exists. Once a user has been selected or created, the messages of the 'Welcome Page' will change . On the main menu page the users are now welcomed with their username.



Single Tap

ONE FINGER DOWN	SYMBOL
○○○○●	e
○○○●○	t
○○●○○	a
○●○○○	l
●○○○○	n
TWO FINGERS TOGETHER	
○○○●●	o
○○●●○	i

ONE FINGER UP	SYMBOL
●●●●○	r
●●●○●	d
●●○●●	c
●○●●●	j
○●●●●	u
ONE FINGER CHASING TWO	
○●●○●	g
●●○●○	x

Figure 4.5: This Figure shows an extract of the *Glossary* view for left-handed user.

Hints. The other *Side Menu Item* is *Hints*. This is to give the user hints on how to avoid and solve difficulties with the application or the Tap Strap. These hints include that the user should set the keyboard layout to US keyboard layout, so that the mapping of the Tap Strap is correct. Another hint is that the user should make sure that the Tap Strap is in the appropriate mode and that the corresponding mapping for the application is used. Otherwise, the users would receive an incorrect character in both cases, even though they have entered the correct combination.

Glossary. Next, the *Main Menu Items* are explained, starting with the *Glossary*. This serves to give the user an overview of the mapping and the associated combination. The overview is subdivided according to the different modes that the Tap Strap has (Single-Tap, Double-Tap, Triple-Tap, Shift, Switch). Depending on whether the user has selected the right or left hand during registration, the *Glossary* is displayed accordingly (Fig. 4.5). The *Glossary* is not only accessible via the main menu, but also in the menus of the *Practice* and *Games Parts*. This should make it easier for the user to briefly look into the *Glossary* if they are unsure about a certain combination.

Practice (Main menu)

The *Practice Part* is divided into the individual modes of the Tap Strap, which are Single-Tap, Double-Tap, Triple-Tap, Shift and Switch. There is also a section on *Further Functions*. The structure of the menus of the individual modes is the same. Within a mode there are several *Learning Units*, in which the user learns 2-5 new characters. How many *Learning Units* a mode has depends on the number of combinations used within the mode. When a session is selected, the user

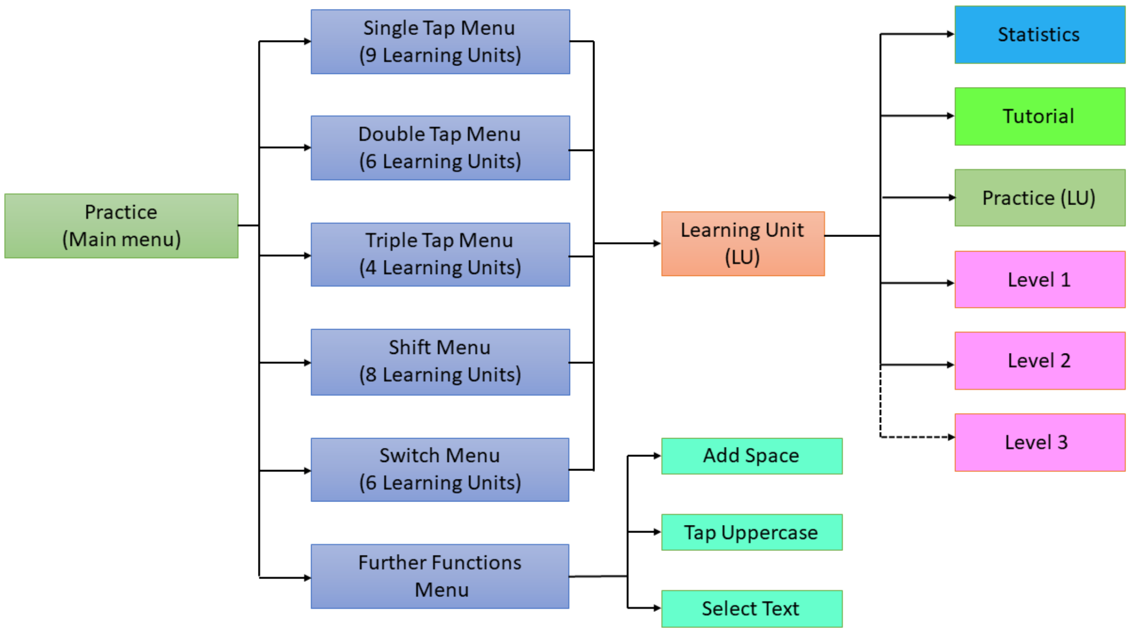
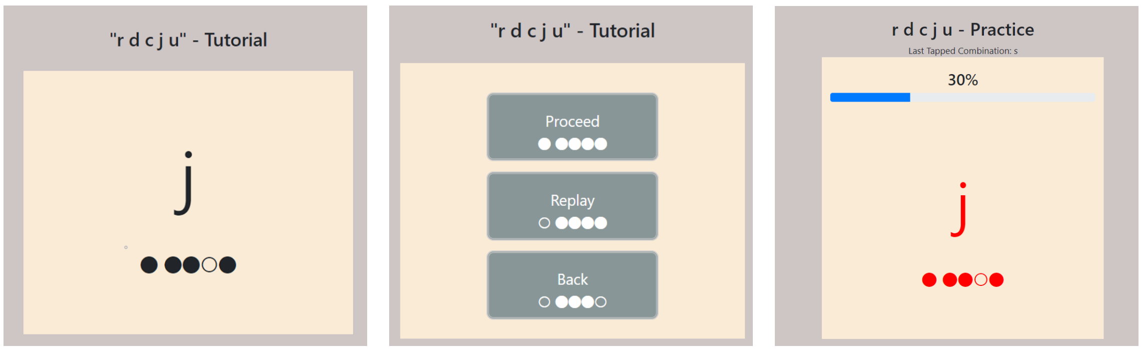

Practice (Main menu)
Single Tap Menu (9 Learning Units)
Double Tap Menu (6 Learning Units)
Triple Tap Menu (4 Learning Units)
Shift Menu (8 Learning Units)
Switch Menu (6 Learning Units)
Further Functions Menu
Learning Unit (LU)
Add Space
Tap Uppercase
Select Text
Statistics
Tutorial
Practice (LU)
Level 1
Level 2
Level 3

Figure 4.6: This Figure shows a map of the structure of the *Practice Part* (Main menu), and the submenus that appear, as well as the structure of the submenus. The dashed line means that *Level 3* is not available in every *Learning Unit*.



(a) 'r d c j u' - *Tutorial*. **(b)** End Dialog *Tutorial*. **(c)** Incorrect tap input.

Figure 4.7: The left Figure (a) shows the view of the *Tutorial* of a *Learning Unit*, the middle Figure (b) shows the end dialog of the *Tutorial*. Figure (c) shows the view of the *Practice Part* when an incorrect tap is made.

sees the menu view of a *Learning Unit*. The *Learning Unit Menu* of a session consists of a statistic showing the accuracy of the characters of the *Learning Unit*, a *Tutorial*, a *Practice* and a *Level Part*. The structure of the *Practice Part* is shown in Fig. 4.6.

Tutorial. The *Tutorial* is used to show the user the combinations of characters that are part of the *Learning Unit*. For this purpose, a character with the corresponding combination is displayed (Fig. 4.7a). After two seconds, the next character with the corresponding combination is shown. In total, there are two runs, i.e. each character is displayed twice. At the end of the *Tutorial*, a dialog appears (Fig. 4.7b) where the user can choose to continue with next part of the *Learning Unit* which would be the *Practice Part*, play the *Tutorial* again or return to the *Learning Unit Menu*.

Practice (Learning Unit). If the user decides to continue with the *Learning Unit* after the *Tutorial*, they will be taken to the *Practice Part*. Here the character are displayed in the same way as in the *Tutorial Part*. The next character appears when the user has tapped the correct combination. Here, the user also has two runs. When the wrong combination is tapped, the character and the combination light up red (Fig. 4.7c). At the end of the *Practice Part* the same dialog appears as in the *Tutorial Part* (Fig. 4.7b). If 'Continue' is selected in this dialog, the user continues with the *First Level* of the *Learning Unit*.

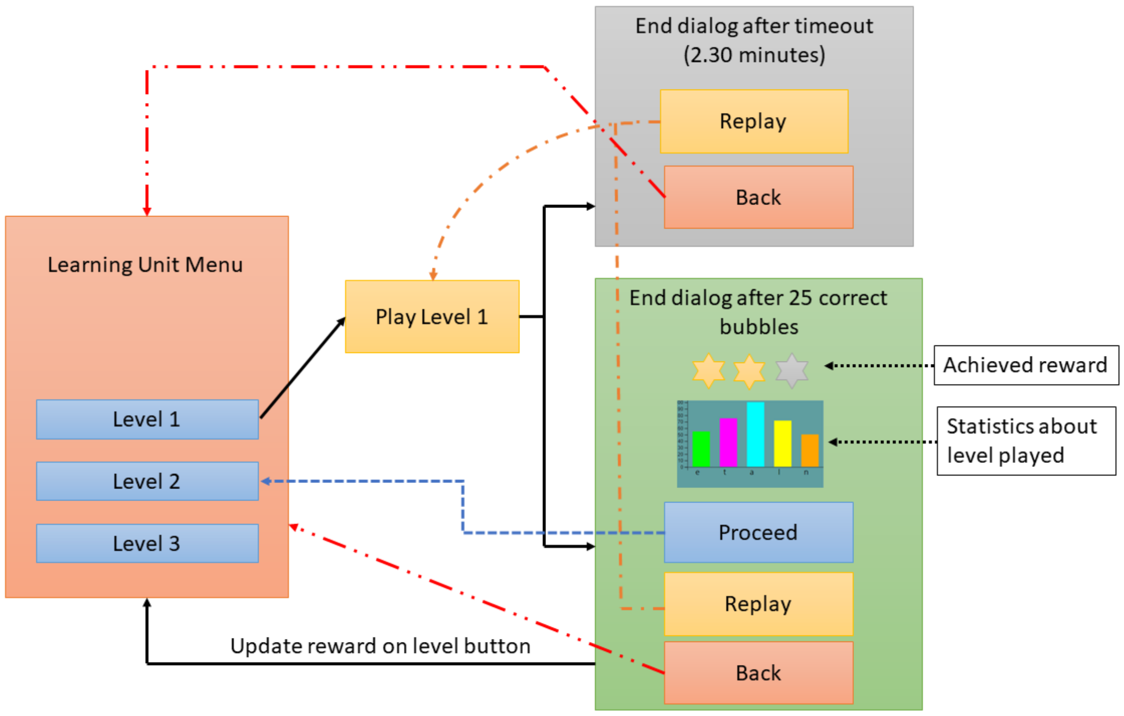
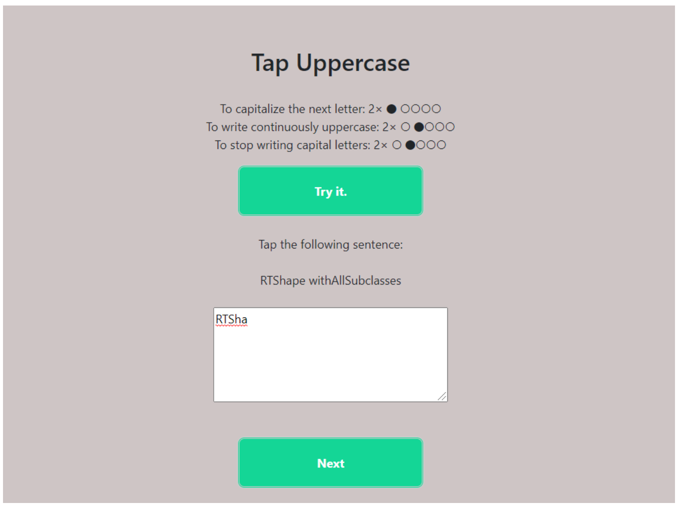

End dialog after timeout (2.30 minutes)
Replay
Back
Learning Unit Menu
Play Level 1
Level 1
Level 2
Level 3
End dialog after 25 correct bubbles
Achieved reward
Statistics about level played
Proceed
Replay
Back
Update reward on level button

Figure 4.8: This figure shows the workflow of *Learn2Tap* when a user selects the first *Level* within a *Learning Unit*.

Level. Fig. 4.8 shows the workflow of the *Levels* (here using the example of *Level 1*), which is also explained below. The first *Learning Unit* of a mode has *2 Levels*, the remaining *Learning Units* have *3 Levels* each. The *Levels* serve to consolidate what has been learned. Within the *Levels* of a *Learning Unit* there are also character from previous *Learning Units*. There are no character from other modes than the one in which the *Learning Unit* is. The *Levels* consist of falling bubbles with a character inside. If the user taps the correct character, the bubble disappears, the progress of the *Level* increases and a bubble with a new character starts to fall. After 25 successfully tapped bubbles a *Level* is finished. After 2.30 min a *Level* is stopped by a timeout. If the user taps the wrong combination twice for a corresponding bubble, the correct combination to be tapped is displayed. The bubble lights up red if the wrong combination is tapped. At the end of a *Level*, the user will see a dialog with the same options as in the *Practice* and *Tutorial Part* (Continue leads to the next *Level*), as well as a statistic about the accuracy of the newly learned characters, of the *Learning*

Unit. When the user achieves more than 55% accuracy, the user gets one star, two stars for 70% accuracy and three stars for 85% accuracy. Achieved stars for a *Learning Unit* are displayed in the menus of the modes and can be found in the menus of the *Learning Units* on the level buttons. The number of the most achieved stars within a *Level* is displayed.


Tap Uppercase

To capitalize the next letter: 2× ● ○○○○
To write continuously uppercase: 2× ○ ●○○○
To stop writing capital letters: 2× ○ ●○○○

Tap the following sentence:

RTShape withAllSubclasses

RTSha

Figure 4.9: This is an example of a *Learning Unit* from the section *Further Functions*.

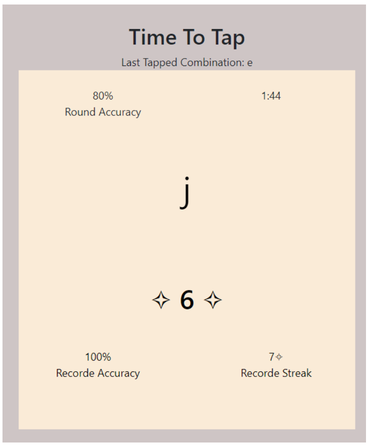
Further Functions. The menu item *Further Functions* (Fig. 4.6) is divided into the sub-items 'Add Space', 'Tap Uppercase' and 'Select Text'. These sub-items are not practiced within their modes, because they are hard to represent by bubble. The tutorials are designed as follows: There is a small explanation of which combination does what and how it is intended to be used. Afterwards, there is a small text which is to be typed. The text contains the newly learned function. Fig. 4.9 shows the tutorial for 'Tap Uppercase'.

Games

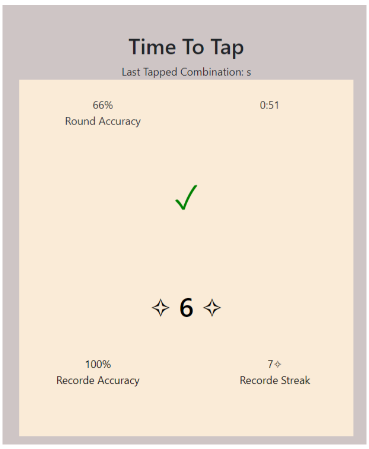
In total there are three games in *Learn2Tap* which are *Time To Tap*, *Tap It 5-Times* and *Coding*.

Time To Tap. The game *Time To Tap* can be played with all or individual Tap Strap modes. With the help of the game, users should find it easier to remember the learned combinations of a mode. The game works as follows: The user is shown characters from the selected modes based on their selection of modes. Once a character is displayed (Fig. 4.10a) the player has one tap to tap the correct combination. If the correct combination is tapped, a green tick appears in place of the character (Fig. 4.10b), the user is credited with a point and the accuracy value increases. If the wrong combination is tapped, a red cross appears in place of the character (Fig. 4.10c), the correct combination is displayed, the player loses one point and the accuracy value decreases. In total, a player has 2 minutes to play the game and collect as many points as possible within these two minutes.

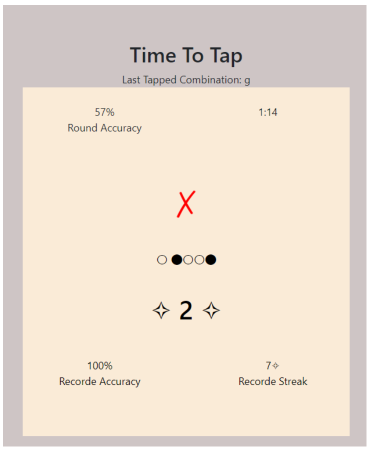
Tap It 5-Times. Next, the game *Tap It 5-Times* is introduced. The game requires the user to know Single -Tap, Double-Tap, Triple-Tap and Switch mode. Here the user is shown short words/code snippets that have to be tapped five times. The goal is that with each tap of the word, the user has a faster and more accurate input. For this purpose, after successful input, for each word tapped, the



(a) *Time To Tap*.

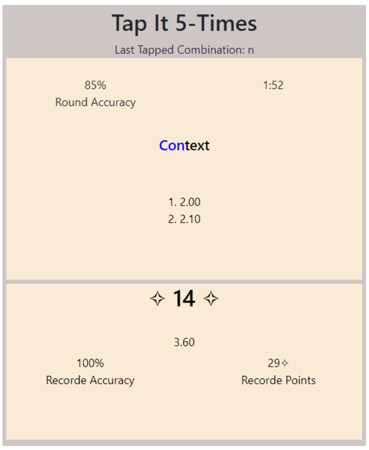


(b) Correct Tap.

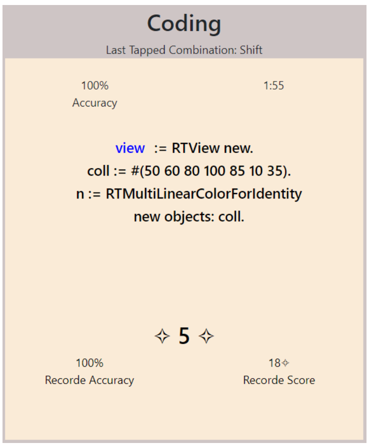


(c) Wrong Tap.

Figure 4.10: The Figures above show the different views during the game *Time To Tap*. (a) Representation of the character to be tapped, (b) the correct character was tapped, (c) a wrong combination was tapped.

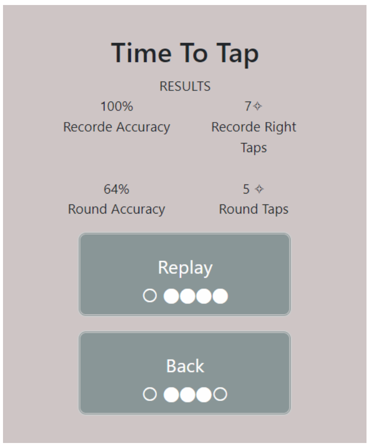


(a) *Tap It 5-Time*.



```
view := RTView new.
coll := #(50 60 80 100 85 10 35).
n := RTMultiLinearColorForIdentity
new objects: coll.
```

(b) *Coding*.



(c) End Dialog Games.

Figure 4.11: Figure (a) shows the view of the game *Tap It 5-Time*, (b) displays the view of the game *Coding*, (c) shows the end dialog shown to the user after a game.

time taken to tap the word is also displayed (Fig. 4.11a). After a word has been successfully tapped in one round, the user receives one point for each character within the word. If the user taps the wrong character twice, the correct combination is displayed, which is required to continue with the next character. If a word/code snippet has been tapped correctly five times, the next word/code snippet is displayed. In total, the game lasts two minutes.

Coding. The last game available in *Learn2Tap* is called *Coding*. This game is designed to prepare the user for programming within AVAR (see section 3.2). For this purpose the user is shown a code snippet from AVAR, which has to be typed (Fig. 4.11b). The ready typed characters within a line are displayed in blue. If the correct character is typed, the user receives one point, if the wrong character is typed, one point is deducted. After two incorrect entries of the character to be tapped, the correct tap combination is displayed, which the user must tap to continue with the next character. To play the game successfully, the user must be able to use the Single-Tap, Double-Tap, Triple-Tap and Switch modes. The player is given two minutes to tap a code snippet. Before the game starts, the user can select how many rounds the game should be played in a drop-down menu. One round consists of one code snippet and lasts two minutes each.

After finishing a game, a dialog displays accuracy and score achieved by the user in the game. Additionally, the record accuracy and record score of the player is displayed (Fig. 4.11c). With the option ’Replay’ the user can play the game again. Pressing the button ’Back’ get the player back to the menu overview of the games.

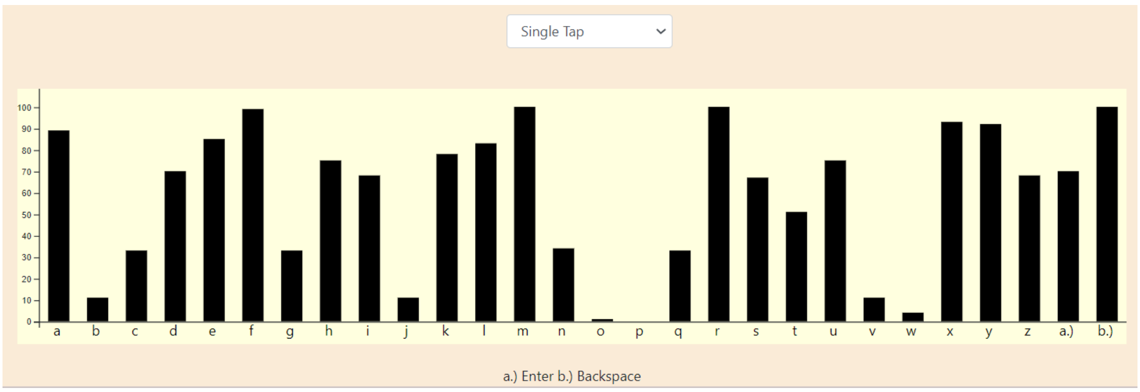


Figure 4.12: This is an example of a Single-Tap mode statistic.

Statistic

The statistics section shows statistics for the individual modes of the Tap Strap and the games that exist within *Learn2Tap*. The user can select the statistic to be displayed via a drop-down menu. The statistics are represented by bar charts. The bar charts of each mode (Fig. 4.12) give information about the accuracy of the characters within the mode. By looking at the statistics, users can identify their weaknesses and repeat certain *Learning Units* accordingly. The bar charts of the games show the results of the last 20 games, where 1 shows the result of the last game played and 20 shows the result of the game furthest back. The left bar shows the user’s record of the game.

4.2.3 Server

The back-end of the web application is implemented as a server. The main task of the server is to establish a connection to the database, read data and write data. Furthermore, the server receives requests from the various clients, which then receive a response from the server. For example, when a user goes to the *Glossary* in the user interface, the client sends a request to the server that the corresponding client needs the data for the *Glossary*. The data for the *Glossary* is stored in the database, i.e. the server executes a corresponding query on the database, receives the response set and sends it back to the client. This way of communicating the queried data occurs frequently in the web application and is listed below:

- Loading the data of an already existing user.
- Loading the data of the *Glossary*.
- Loading the menus of the individual modes in the *Practice Part* (query of the characters that belong to a *Learning Unit*, as well as the reward received so far for the individual *Learning Units*).
- Within the *Learning Units*, loading the statistics, as well as the reward for each *Level* within the *Learning Unit*.
- Loading the characters and tap combinations that occur within a *Learning Unit* in the *Tutorial*, *Practice* and *Level Part*.
- Loading the characters and code snippets that occur within the games.
- Query of statistics values within the statistics area.

Another task of the back-end is to write the data sent by the client. For example when creating a new user. In this case, the client first sends the request to check whether the username already exists. If this is not the case, a user with the entered name is created, and all tables that exist to manage the learning status of the users add the new user. Further write operations of the back-end to the database occur at the following events:

- Creating a new user.
- Finishing a *Level*. Sending to the back-end the reward received as well as the accuracy of the characters shown within the *Level*.
- Sending the accuracy data after finishing a game, as well as the points received in the game.
- Sending user tracking data. These consist of: The users interactions with the learning application, the tapped keys during a game or *Level*, the typing speed of tapped keys and and the result of the game/*Level* (statistics, reward, score).

4.2.4 Database

The database is the final part of the learning application and runs on the same local machine or server where the back end is installed. This database is used to store user data and progress. Other contents are the mapping of the tap combination to the characters, the division of the *Learning Units* into the different modes and the code snippets used within the games. As explained in section 4.2.3, the database sends and receives the data from the back-end. In total there are 18 tables in the database schema for *Learn2Tap*. These are divided as follows:

- Ten tables are used to store the learning status in each mode (Single-Tap, Double-Tap, Triple-Tap, Switch, Shift). Each mode has two tables, one stores the users reward for each *Level* of the *Learning Units*. The other table stores the accuracy of the characters present in the mode.
- Each of the three games has a table that stores the points obtained in the last 20 games, as well as the records of accuracy and points obtained.
- The table *gamesentence* contains the code snippets that appear in the games *Tap It 5-Times* and *Coding*.
- The mapping of the combinations to the different characters in the different modes is shown in another table.
- A further table contains the distribution of the characters for the different *Learning Units* of a mode.

Package	Version	Function
General		
Node.js [1]	^14.16.1	JavaScript run time environment that handles asynchronous event requests.
Client		
react[2]	17.0.2	Developing user interface.
react-router-dom [5]	^5.2.0	Enables routing in React.
react-dom[6]	^17.0.2	Loads React elements into a provided container. Here in the '<script>' tag of the index.html.
js-cookies[7]	^2.2.1	Authorizes setting and reading of cookies.
axios[8]	^0.21.1	Sending requests from the client to the server, as well as receiving responses from the server.
d3[9]	^6.7.0	Used to create barcharts.
Server		
cors[10]	^2.8.5	Enables communication between the port of the client and the server.
express[3]	^4.17.1	Configuring and managing an HTTP server.
MySQL[4]	^2.18.1	Provides a MySQL driver, which allows communication with the database.
nodemon[11]	^2.0.7	Automatic reloading of the node.js application when the code changes.

Table 4.1: The table gives an overview of the software stack used in the implementation of *Learn2Tap*.

- The table *user* stores the registered users, as well as their selected hand.
- The last table is used to store the user interactions, which are for example the tapped keys during a game or *Level* and the typing speed of the tapped keys.

4.3 Implementation

This section presents the implementation of *Learn2Tap* in detail. Node.js[1], React.js[2], express[3], and MySql[4] provide the main foundation of the web application. Node.js is a JavaScript run time environment that handles asynchronous event requests, making the application scalable and allowing multiple clients to send requests to the server in parallel. Express is a Nodes.js framework that provides powerful features and functionality to facilitate the creation of a web application. The Java Script library React.js was used to create the user interface. MySql was chosen as the database. Detailed information about the implementation of the client, server and database is given below.

[1]https://nodejs.org/en/, accessed 06.09.2021
[2]https://reactjs.org/, accessed 06.09.2021
[3]https://expressjs.com/de/, accessed 06.09.2021
[4]https://www.mysql.com/de/, accessed 06.09.2021
[5]https://www.npmjs.com/package/react-router-dom, accessed 06.09.2021
[6]https://www.npmjs.com/package/react-dom, accessed 06.09.2021
[7]https://www.npmjs.com/package/js-cookie, accessed 06.09.2021
[8]https://www.npmjs.com/package/axios, accessed 06.09.2021
[9]https://www.npmjs.com/package/d3, accessed 06.09.2021
[10]https://www.npmjs.com/package/cors, accessed 06.09.2021
[11]https://www.npmjs.com/package/nodemon, accessed 06.09.2021

Listing 4.1 This listing shows the structure of the routing system and the use of cookies (index.js).

```
//------------------html file----------------------------------------
 <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
 </body>

//-----------------index.js------------------------------------------
import{Welcome} from './App/UserAdministration/Welcome';
import {Glossary} from "./App/overview/glossary";

function Index(){
  const [value,setValue] = useState(0);
  const providerValue = useMemo(( )=> ({value, setValue}), [value,setValue]);


  var valIdCookie = Cookies.get('userID');
  if(value !== 0){
      Cookies.set("userID",value, { expires: 7 });
  }else{
    setValue(valIdCookie);
  }

  return(
      <div className= "Index">
          <Switch>
            <idContext.Provider value={{value,setValue}}>
                <Route exact path="/Learn2Tap/UserAdministration" component= {Welcome} />
                <Route exact path="/Learn2Tap/Glossary" component= {Glossary} />
            </idContext.Provider>
         </Switch>
      </div>
  )
}

const rootElement = document.getElementById("root");
ReactDOM.render(<BrowserRouter> <Index/> </BrowserRouter>, rootElement);
```

4.3.1 Client Application

In this section, I explain the implementation of the front-end. First, I give an explanation of the software stack used and its purpose. Then four key concepts are presented, these are routing which allows the inclusion of multiple sub-websites. Then the structure of the sub-websites is explained, as well as the concept of the Class Component which allows components like the bar charts to be defined as a class. The last key concept is style sheets. The section is concluded by explaining the workflow of 'Main Menu → Double-Tap Menu → Learning Unit Menu → Level'.

Software Stack

A summary of the packages used within the client can be found in Table 4.1. React.js was used as the basis for implementing the interface. An extension of React.js is the package react-dom, which provides a container for the different React elements. The container is located inside the `<script>` tag of an html file and is addressed by ID (see listing 4.1 line 2-5,35-36). The package 'react-router-dom' allows routing within React, since React is actually used to develop single-page applications. When calling another page, within the web application, it is not possible to pass variable values, therefore cookies were used to pass the variable values. The reading and write of cookies are provided by the package 'js-cookies'. Communication with the server was implemented by the package 'axios', details about client-server communication can be found in subsection 4.3.2. For the visualisation of data (here the bar charts and reward stars) D3 was used. The paragraph 'Class Components' describes in detail how to visualise the reward stars.

Key Concepts

This subsection explains the key concepts, which are used within the client of *Learn2Tap* (the communication with the server is excluded, see section 4.3.2). The key concepts are, firstly routing, which allows to create a website with routing from the actual single page website. For this purpose the different website views are created by means of 'Const Components'. Within the 'Const Components' there are Class Components which allow to import frequently used graphics (bar charts, reward stars) by means of classes. As last key concept, the 'Style sheets' are explained, in which the design of the user interface is defined.

Routing The routing is implemented in the clients 'index.js'. This file is automatically initialised when a React project is created. Inside this file it is defined what will be rendered in the application. The individual components from the various files, such as 'Glossary', 'Coding' or 'Level', are imported into the index.js file by means of `import` statement (lst. 4.1, line 8,9). The routing system to the individual files is specified in the `return()` function between the `<switch>` tag. For example, the structure of a single route looks like this: `<route exact path="/Learn2Tap/UserAdministration " component= {Welcome` />, where `path='...'` specifies the URL path to the component. Using `componente = ...` the corresponding component is linked (lst. 4.1, line 27) . The `<...Context. Provider ...>` tags provide variables that can be accessed from all files that import the corresponding provider (lst. 4.1, line 12,13,26,29), but only after the component has been rendered for the first time. With the command `document.getElementById("root");` the container inside the html file is accessed. `ReactDOM.render(<BrowserRouter> <Index/> </BrowserRouter>, rootElement);` (lst. 4.1, line 35,36) is used to specify what should be rendered.

Structure of the Sites (Const Components) All websites that can be called by their own `Learn2Tap/...` path (e.g. all menus or the *Glossary*) are structured as components as follows (see listing 4.2): The components have the following name `export conts nameOfComponent = props =>...` (line 3), where at the place where `nameOfComponent` stands, the actual identifier of the components is entered. Within the body of the components are the variables (as `useState` (line 6)) and the functions (line 8 - 48) realised. At the end of each component is a return statement (line 50 - 54). Inside this return statement is a mixture of html and react code, which defines the view that will be rendered.

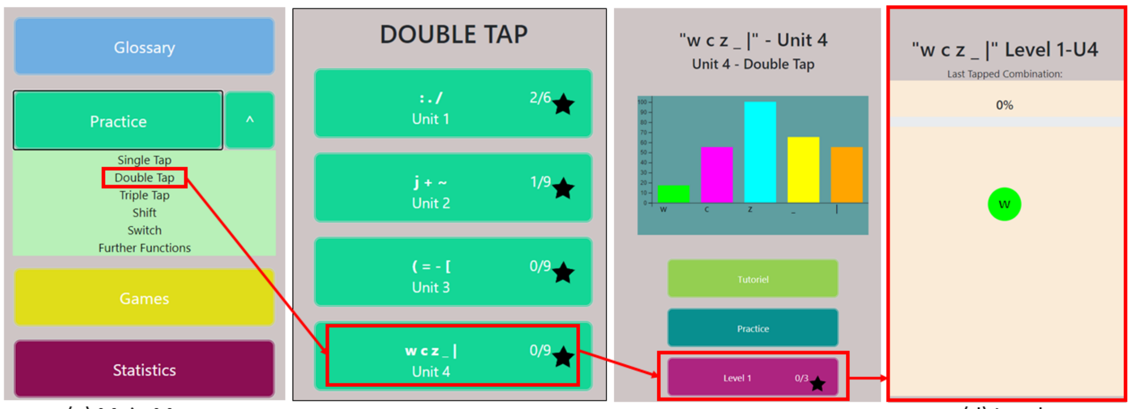
Listing 4.2 Component Structure.

```
import {idContext} from "../../idContext";

export const Practice = props => {

  const {value,setValue} =  useContext(idContext);      //use Values from Context Provider
  const[initialize,setInitialize] = useState(true);

  #defining all states & functions

  //key press function
    function useKeyPress() {
      // Checks whether a key has been pressed
      const [keyPressed, setKeyPressed] = useState(false);
      // With key press call of the function
      function downHandler({ key }) {
        //...
      }

      // If released key set to false
      const upHandler = ({ key }) => {
        if (symbolArr.indexOf(key) !== -1) {
          setKeyPressed(false);
        }
      };

        // Event listener
        useEffect(() => {
          window.addEventListener("keydown", downHandler);
          window.addEventListener("keyup", upHandler);

          return () => {
            window.removeEventListener("keydown", downHandler);
            window.removeEventListener("keyup", upHandler);
          };
        }, );
        return keyPressed;
      }

  //Defining the timer
      useEffect(() => {
        const timer = setInterval(() => {
            //....
          }
         }, 100);

   //clearing interval
        return () => clearInterval(timer);
      });

    return(
  //html/react code,defines the view
      <div id="returnMenuDiv">
      <!---- ....--->
      </div>
    );
};
```

The code above the return statement is mostly inside functions, so that the code is not executed in a continuous way. To implement the learning application two essential functions were used, which were implemented by `useEffect`. One of the functions is the setting of an interval, by using a timer (lines 40 - 44). This way functions can be executed at any time, e.g. after 2 minutes or when a certain event occurs. The other important function is the `useKeyPress()` (line 11 - 37) function which is implemented using event listener. When a key is pressed this is realised within the function by `downHandler(key)`. Depending on whether the keypress is correct or not certain functions are executed. The keypress function only occurs within the components where user input is required.

Class Components React components that are defined as classes have the advantage that they can be called and rendered with variable values as often as you like by using tags (`<nameOfClassComponent ></nameOfClassComponent>`) within the return method of component files. In *Learn2Tap*, these Class Components are used to create d3.js visualisations. This is the case for bar charts and the reward stars. Each of these visualisations has two class React.Componente, where the classes `callBarchart` and `callStar` are used to pass the variable values from the actual files. The components are then drawn in the classes 'D3.barchart.js' and 'starComponent.js'. In the second files the graphics are drawn into an SVG using the `drawChart()` function. This function is executed for each class call once at the beginning. The sequence of such a call is shown in listing 4.3 for the starComponent. It was not possible to call the files 'D3.barchart.js' and 'starComponent.js' directly from the components, because the passing of parameters did not work. Therefore, the inter-class components (`callBarchart` and `callStar`) had to be created.

Style sheets The css stylesheets are used to define the design of the learning application. The individual elements are addressed by ID or classNames. This only applies if the corresponding css file is imported into the component with the IDs and class names to be addressed. With the help of the css files the learning application becomes responsive because the design adapts to the different display sizes.



(a) Main Menu. (b) Double-Tap Menu. (c) Learning Unit Menu. (d) Level.

Figure 4.13: This Figure represent the course of the 'Workflow' example.

Listing 4.3 Shows the code needed to use a class component.

```
//--------------------------component file--------------------------------
 <CallStar colorF={colStarOne} colorS={colStarTwo} colorT={colStarThree}>
 </CallStar>

//-------------------callStar.js-----------------------------------------
class CallStar extends  React.Component{
  state = {
        colorF: this.props.colorF,
        colorS: this.props.colorS,
        colorT: this.props.colorT
    }

      render() {
        return (
           <div>
        <Star
          colorFirst={this.state.colorF}
          colorSeconde={this.state.colorS}
          colorThird = {this.state.colorT}>
        </Star>
      </div>
        )}
  }
export default CallStar;

//---------------starComponent---------------------------------------
class Star extends React.Component {
        componentDidMount() {
                this.drawChart();
              }

    drawChart(){

    //Code which draws stars in the SVG
    }
    render(){
     return (
      <div className="showStar">
         <svg id='starSVG' background-color="blue" ></svg>
       </div>
     )}}
export default Star;
```

Workflow

This subsection uses an example to explain the workflow from calling up a mode menu of the *Practice Part*, to selecting a *Learning Unit*, until playing the *Level*, showing which processes are executed in the client during this time. Fig. 4.13 shows the course of the example.

Main menu The starting point of the example is the view of the main menu (Fig. 4.13a), which is defined in the file 'menu.js' and can be reached via the URL `http://localhost:3000/Learn2Tap`. Before a user can start practicing something in the *Practice Part*, a user must be registered. This information is requested by the package 'js-cookies' (see table 4.1). It is queried whether the cookie 'userID' is set for the website (see listing 4.1, line 16 - 20). If a cookie is set for the 'userID', the user data (hand and name) are retrieved via back-end request. If the user clicks on the practice button the links to the different mode menus are loaded. When clicking on a link, the user is redirected to the corresponding mode menu. In the case of the example, it is the menu of the Double-Tap mode.

Double-Tap Menu When the 'Double-Tap Menu' (Fig. 4.13b) is accessed, the cookie 'TypingMode' is set first (here: doublTap), so later in the 'Workflow' the other 'Files' can query for which mode data must be requested. Which *Learning Unit* contains certain characters and how many stars the user has achieved in this *Learning Unit* so far is stored in the database. Therefore, a corresponding query is executed using 'axios[8]'. Afterwards the answers are stored in the corresponding constants. In this example, the user selects the fourth session. When a button is selected, the following actions are performed (see Appendix listing A.1): The function `sendUserTracking( userID, event, eventName , location)` is executed. That function sends to the back-end the information which action the user has just selected. The back-end passes this information to the database, creating a corresponding entry in the *userTracking* table. Then the function `getLevelValues(symbolIds, rightHand, currModus, currUnit, unitID, firstUrl)` is called, and the cookie for the 'symbolIdsArray' is set. The IDs being set correspond to the 'symbolIds' of the symbols from the *level* table in Double-Tap mode. For example, symbol 'w' in Double-Tap mode has ID 15 and the 'c' has ID 17 (compare table 4.16).

The information 'Tap Mode', 'Unit Name', 'Unit Id' and 'Symbolids' are set as cookies within the `getLevelValue(...)` function, to have this information available in the next menu (*Learning Unit Menu*). By means of a back-end query, the following data are determined: the characters, tap combinations and key combinations of the *Learning Units*, as well as characters, tap combinations and key combinations of all characters that have been practiced within the mode so far (i.e.. concretely in this example all characters of *Learning Unit 1 - 3*). The answer arrays are set as cookies. Furthermore, cookies are used to determine whether the menu of the *Learning Unit* has 2 or 3 *Levels* to load the corresponding view.

Menu Learning Unit The menu of the *Learning Units* (Fig. 4.13c) with three *Levels* is stored within the 'menuLU3Level.js' and accessible via the URL `http://localhost:3000/Learn2Tap/MenuPracticeUnit3`. The flow of the files of the *Learning Unit Menus* with two or three *Levels* is identical, with the difference that the *Learning Units* with three *Levels* have one button more. When a user visits the menu of a *Learning Unit*, the cookies 'TypingMode', 'unitId', 'symbolArrCookie' and 'symbolID ' are read first. If a character from the character array has a length greater than five, this is represented on the x-axis of the bar chart by a placeholder (e.g.. 'a.)'). Fig. 4.14

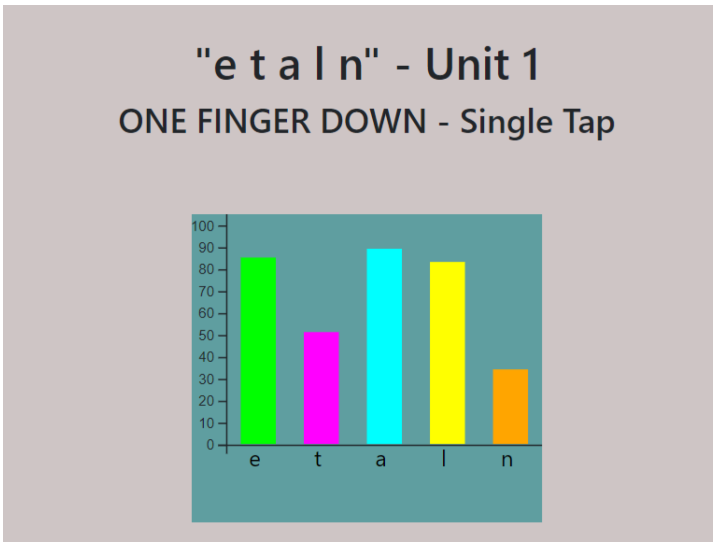


Figure 4.14: This Figure shows an example of the presentation of the statistics of a *Learning Unit*.

shows the statistics within a *Learning Unit* without placeholders. Depending on the typing mode (here Double-Tap), the stars as well as the statistics are requested for the according mode with the corresponding *Learning Unit*. After receiving the answer from the server, the related constants are set. Once the initialisation of the learning menu is complete, the user can see the menu. By clicking the button 'Level 1', a cookie with the number of the *Level* will be set. Another cookie stores the number of *Levels* the *Learning Unit* has (here 3). Next, the user tracking function is sent and finally the user is redirected to the selected *Level*.

Level The Level website (Fig. 4.13d) has the URL `http://localhost:3000/Learn2Tap/Level`. First, the *Level* is initialised with the data of the corresponding *Learning Unit*. This includes the selected hand of the user, the typing mode, the unit ID and the unit name (here Unit 4). The cookies are then queried, which contain the characters of the *Learning Unit* and the characters of the previous *Learning Units*. These are used to determine the length of the array that store how many times a character appeared, how many times the character was tapped correctly and how many attempts the user needed until the character was tapped correctly. Depending on the typing mode, the following data is obtained from the back end: Characters, combinations and key combinations of the *Learning Unit*, as well as previous characters, combinations and key combination of the previous *Learning Units* of the mode. In this case, the key combination corresponds to all the keys that are output when tapping a character in 'Double-Tap Mode' (see explanation section 4.2.1). It is not possible to read the data, for example of the key combinations, from the cookies, as the content is formatted by setting them as cookies. After receiving all data, the first bubble is initialised. Within the *Level* file a timer runs continuously, which executes its body once every second. A *Level* can assume 3 modes after initialisation, these are 'countdown', 'play' and 'pause'. Depending on the mode the body of the timer is executed. In 'countdown' mode, the user is shown a countdown that counts down from 3. Then the *Level* continues or is started. In 'pause' mode the user will see the pause dialog.

In the following, the 'play' mode of the *Levels* will be described in more detail since this is the main function of the Level component. Besides the interval function, there are eleven other functions.

calculateStatistic() This is the function used for calculating the statistics displayed in the end dialog. The following loop is used to calculate the statistics for the characters to be learned in the *Level*. Depending on the total accuracy of these characters, the user receives golden stars (55% = 1 star, from 70% = 2 stars, from 85% = 3 stars). Furthermore, the method calculates the statistics about all appeared characters and sends them to the corresponding function in the back-end depending on the mode. Furthermore, the following information is sent via the user tracking: All tapped characters, the order in which the characters were in the bubble, the typing speed for a character, the reward, as well as the statistics about the characters within the *Level*.

nextShiftKeyCombi(combis, currLetterCount) This function creates an array for the current character in the bubble, which corresponds to the number of keys that appear in the tap combination (see Appendix lst.A.2). Furthermore an array is created which stores whether all characters have appeared, i.e. this array has the same length as the one with the tap combinations, whereby all entries with the value false are initialised. This function is called in the function to create a bubble.

createShiftObject(keyPressed) If the level is not played in Single-Tap mode, this function is used to check that all keys that must be tapped for a character have been tapped. This function is called within the `keypress` function. If the pressed key is within the key combination to be tapped, the array which stores whether a certain key of the combination was tapped at this point is set to true. In the other case, if the key is not contained in the combination, the whole array is set to false again. Within the interval of the timer, it is checked in each run whether all keys have been tapped (i.e.. the array which stores whether all corresponding keys have been tapped contains only the value 'True'). If this is true, the function to create a new bubble is called.

useKeyPress() This method (see listing 4.2, line 10 - 37) executes functions depending on the game mode (pause dialog, play or end dialog). The functions in play mode are the most important. If the *Level* is not played in Single-Tap mode, a keyPress is passed to the function `createShiftObject(keyPressed)`. In Single-Tap mode it is checked if the key input corresponds to the character on the bubble. When this is correct, the function `newBubble()` is called. In case the key input is wrong, the bubble lights up red and the function `wrongHit()` is called. In parallel, the number of times a wrong input has been made for the current bubble is counted. Once this number is greater than two, the tap combination for the character is displayed.

4.3.2 Interface Client-Server

The node.js package axios[8] was used to realise the client-server interface (details table 4.1). Axios enables the client to send requests to the server. Depending on the implementation, the client waits with further program execution until the client has received a response from the server (listing 4.4). The server sends a response using the command `res.send([])` (listing 4.4, line 26). The client can read the response with the function `response.data[0]` (listing 4.4, line 6). Alternatively, the client can continue with the program execution as soon as the request has been sent (listing 4.5), in this case the server does not send a response to the client, or the response is not retrieved. In *Learn2Tap*

Listing 4.4 Example code for communication between server and client using `Axios.get()` function.

```
 1 //----------------Client-----------------------------------------------------------------
 2
 3  Axios.get(`http://localhost:3001/getGlossary/${currHand}`,).then((response)=>{
 4
 5     //Executed after receiving the response from the server
 6     var res  =response.data[0];
 7     setOvervewObject(res);
 8     setInitialize(false);
 9     setCallFunction(true);
10      });
11
12 //----------------Server--------------------------------------------------------------------
13
14 app.get('/getGlossary/:currHand', (req, res)=>{
15
16    //Access to variables from the header
17    const hand = req.params.currHand;
18
19    dbUser.query("SELECT id, righthand, lefthand, single, doubletap, triple, switch, shift
   FROM levels ",  (err, result) => {
20     if(err){
21      console.log(err);
22     }else{
23      const resultJson = Object.values(JSON.parse(JSON.stringify(result)));
24
25      //Sending the response to the client
26           res.send([resultJson]);
27     }
28    })
29   });
```

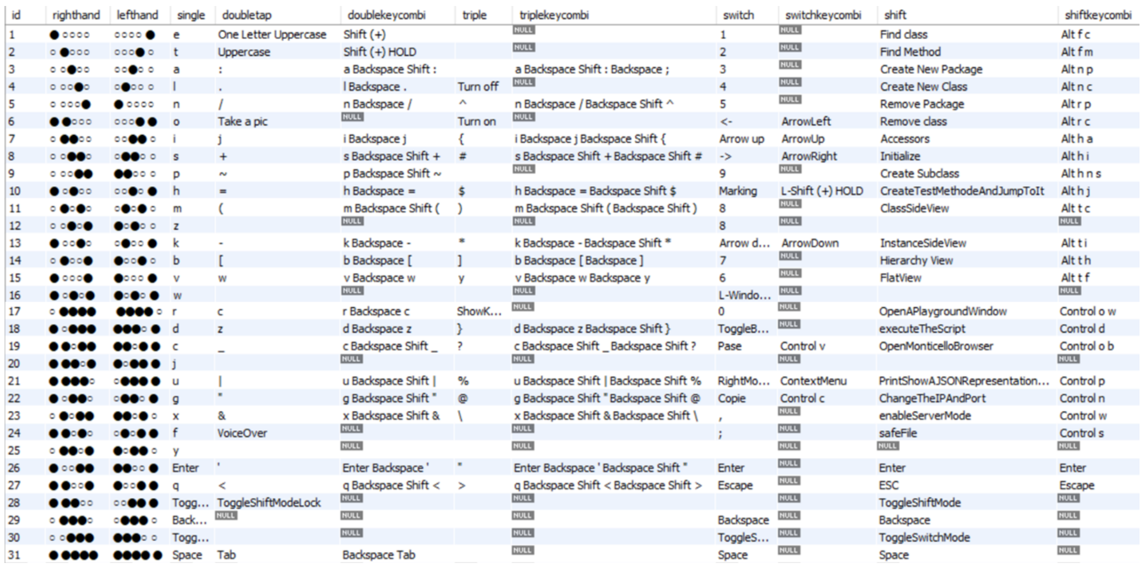
both types of requests (waiting for the response from the server and program continues running) are implemented.
In *Learn2Tap* two types of functions are used for communication between client and server. The `Axios.post` (listing 4.5) and `Axios.get` (listing 4.4) function. The assignment of a request in the back-end is realised by the URL sent by the client (listing 4.5, line 14). The `Axios.post` function was mainly used when sending user data to the server, e.g.. to pass the accuracy of a *Level* to the back-end. With this function, the data is sent within a body and is easily accessible in the back-end. The `Axios.get` function was primarily used to send requests to the server where a response is expected from the server. The variables to be passed are in the header of the request (listing 4.4, line 3). The variables from the header are rather small compared to those from the body of the `Axios.post` function. Examples of variable sent via the header are the 'userID' and the ID of the *Learning Unit*.

Listing 4.5 Example code for communication between server and client using `Axios.post()` function.

```
1  //----------------Client----------------------------------------------------------------
2      Axios.post('http://localhost:3001/statisticSingleTap',
3  
4      {
5      id: userID,       //Variables Values that are sent
6      symbols:symbols,    //to server by means of the function body
7      statistic:statistic,
8      unitLevelID: unitLevelID,
9      stars:stars
10    })
11 
12 
13 //----------------Server-------------------------------------------------------------------
14 app.post('/statisticSingleTap',(req,res) =>{
15 
16   //Reading the variables stored in the body of the function
17     const id = req.body.id;
18     const letter = req.body.symbols ;
19     const resultStat = req.body.statistic;
20     const unitLevelID = req.body.unitLevelID;
21     const stars = req.body.stars;
22 
23   //Database request
24     dbUser.query('SELECT * FROM singletab WHERE userID=?',  id , (err, result) =>{
25         if(err){
26             console.log(err);
27         }else{
28            ...
29        //This part does not contain a res.reponse() function.
30 
31     }
32      })
33 })
```

4.3.3 Interface Server-Database

The interface of Server-Database is realised by the node.js[1] package MySQL[4] (see also table 4.1). This provides a MySQL driver within the node.js application. Before read and write operations can be performed on the database, a connection to the MySQL server of the applications host must be established. This connection is established using the listing from the appendix A.3. The following data is required to establish a successful connection: Host, user, password of the MySql server, and the name of the database. The database commands can then be executed as in the listing from the appendix 4.5 (line 24) and listing 4.4 (line 19), i.e. the queries have a similar structure as normal MySQL queries, except that variable query values such as the ’userID’ or data to be written to the database are realised using placeholders (here ’?’). The placeholders are then replaced by variable values.



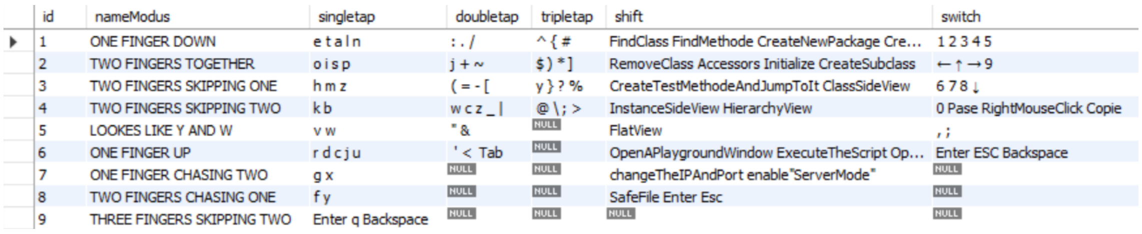
id	righthand	lefthand	single	doubletap	doublekeycombi	triple	triplekeycombi	switch	switchkeycombi	shift	shiftkeycombi
1	●○○○○	○○○○●	e	One Letter Uppercase	Shift (+)		NULL	1	NULL	Find class	Alt f c
2	○●○○○	○○○●○	t	Uppercase	Shift (+) HOLD		NULL	2	NULL	Find Method	Alt f m
3	○○●○○	○○●○○	a	:	a Backspace Shift :		a Backspace Shift : Backspace ;	3	NULL	Create New Package	Alt n p
4	○○○●○	○●○○○	l	.	l Backspace .	Turn off	NULL	4	NULL	Create New Class	Alt n c
5	○○○○●	●○○○○	n	/	n Backspace /	^	n Backspace / Backspace Shift ^	5	NULL	Remove Package	Alt r p
6	●●○○○	○○○●●	o	Take a pic	NULL	Turn on	NULL	<-	ArrowLeft	Remove class	Alt r c
7	○●●○○	○○●●○	i	j	i Backspace j	{	i Backspace j Backspace Shift {	Arrow up	ArrowUp	Accessors	Alt h a
8	○○●●○	○●●○○	s	+	s Backspace Shift +	#	s Backspace Shift + Backspace Shift #	->	ArrowRight	Initialize	Alt h i
9	○○○●●	●●○○○	p	~	p Backspace Shift ~		NULL	9	NULL	Create Subclass	Alt h n s
10	●○●○○	○○●○●	h	=	h Backspace =	$	h Backspace = Backspace Shift $	Marking	L-Shift (+) HOLD	CreateTestMethodeAndJumpToIt	Alt h j
11	○●○●○	○●○●○	m	(	m Backspace Shift (	)	m Backspace Shift (Backspace Shift)	8	NULL	ClassSideView	Alt t c
12	○○●○●	●○●○○	z		NULL		NULL	8	NULL		NULL
13	●○○●○	○●○○●	k	-	k Backspace -	*	k Backspace - Backspace Shift *	Arrow d...	ArrowDown	InstanceSideView	Alt t i
14	○●○○●	●○○●○	b	[	b Backspace [	]	b Backspace [Backspace]	7	NULL	Hierarchy View	Alt t h
15	●○○○●	●○○○●	v	w	v Backspace w	y	v Backspace w Backspace y	6	NULL	FlatView	Alt t f
16	●○●○●	●○●○●	w		NULL		NULL	L-Windo...	NULL		NULL
17	○●●●●	●●●●○	r	c	r Backspace c	ShowK...	NULL	0	NULL	OpenAPlaygroundWindow	Control o w
18	●○●●●	●●●○●	d	z	d Backspace z	}	d Backspace z Backspace Shift }	ToggleB...	NULL	executeTheScript	Control d
19	●●○●●	●●○●●	c	_	c Backspace Shift _	?	c Backspace Shift _ Backspace Shift ?	Pase	Control v	OpenMonticelloBrowser	Control o b
20	●●●○●	●○●●●	j		NULL		NULL		NULL		NULL
21	●●●●○	○●●●●	u	|	u Backspace Shift |	%	u Backspace Shift | Backspace Shift %	RightMo...	ContextMenu	PrintShowAJSONRepresentation...	Control p
22	●○●●○	○●●○●	g	"	g Backspace Shift "	@	g Backspace Shift " Backspace Shift @	Copie	Control c	ChangeTheIPAndPort	Control n
23	○●○●●	●●○●○	x	&	x Backspace Shift &	\	x Backspace Shift & Backspace Shift \	,	NULL	enableServerMode	Control w
24	●●○●○	○●○●●	f	VoiceOver	NULL		NULL	;	NULL	safeFile	Control s
25	○●●○●	●○●●○	y		NULL		NULL		NULL	NULL	NULL
26	●○○●●	●●○○●	Enter	'	Enter Backspace '	"	Enter Backspace ' Backspace Shift "	Enter	NULL	Enter	Enter
27	●●○○●	●○○●●	q	<	q Backspace Shift <	>	q Backspace Shift < Backspace Shift >	Escape	NULL	ESC	Escape
28	●●●○○	○○●●●	Togg...	ToggleShiftModeLock	NULL		NULL		NULL	ToggleShiftMode	NULL
29	○●●●○	○●●●○	Back...	NULL	NULL		NULL	Backspace	NULL	Backspace	NULL
30	○○●●●	●●●○○	Togg...		NULL		NULL	ToggleS...	NULL	ToggleSwitchMode	NULL
31	●●●●●	●●●●●	Space	Tab	Backspace Tab		NULL	Space	NULL	Space	NULL

Figure 4.15: The table *level* stores the representation of the tap combination, as well as which tap combination in a mode results in which character. It also stores all the characters that are output in the background when a character is tapped.

Interface limitations The interfaces have reached their limits in the implementation of user tracking. The plan was to send each tap with the corresponding data from the client to the server. The server would then write this information into the database. This did not work, some of the data was lost and not written to the database. The same problem occurred at the end of a *Level* or game when, for example, four `userTracking` functions were sent in succession. The solution to this problem is to send the next user tracking only after a `reponse` to the previous function has been received.

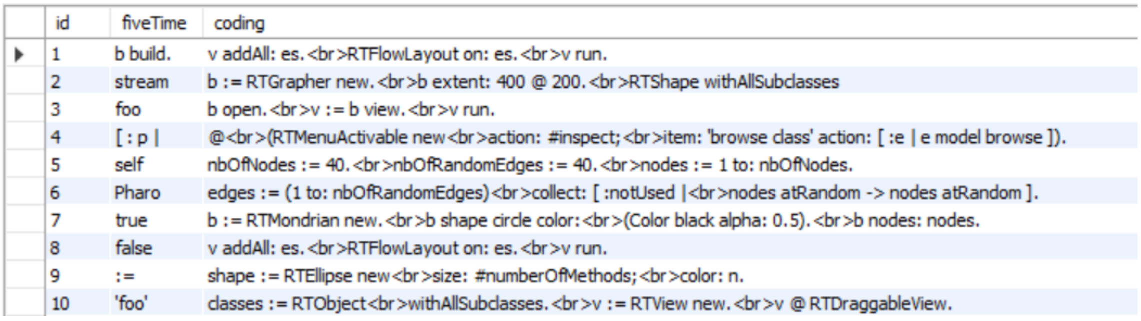
4.3.4 Server

Communication between the server and the front-end is done via express[3], so the server must connect to a port. How this works is shown in a listing in the appendix A.4. In addition to writing and reading data from the database, as well as processing client requests, the Server's task is to prepare data in such a way that it can be written to the database or passed on to the client. When data of the database is to be updated, the old data must first be read from the database and then be offset against the new data that the server has received from the client. This is necessary, for example, when ending a game. In this case the accuracy of the shown characters must be recalculated (see Appendix A.5). The new accuracy is calculated as follows: $newAccuracy = (databaseAccuracy + 2 * clientAccuracy)/3$. If the menu of a *Learning Unit* is called in the front-end e.g.. 'Unit 2' - Double-Tap, then a request is sent to the server, how many stars the user gets for the *Levels* within the *Learning Unit*. The server modifies this query into a MySQL query, and then reads the part of the response data that has the information about the reward of the *Learning Unit*. Afterwards, the server sends the filtered data to the front-end.



id	nameModus	singletap	doubletap	tripletap	shift	switch
1	ONE FINGER DOWN	e t a l n	: . /	^ { #	FindClass FindMethode CreateNewPackage Cre...	1 2 3 4 5
2	TWO FINGERS TOGETHER	o i s p	j + ~	$) *]	RemoveClass Accessors Initialize CreateSubclass	← ↑ → 9
3	TWO FINGERS SKIPPING ONE	h m z	(= - [	y } ? %	CreateTestMethodeAndJumpToIt ClassSideView	6 7 8 ↓
4	TWO FINGERS SKIPPING TWO	k b	w c z _ |	@ \ ; >	InstanceSideView HierarchyView	0 Pase RightMouseClick Copie
5	LOOKES LIKE Y AND W	v w	" &	NULL	FlatView	, ;
6	ONE FINGER UP	r d c j u	' < Tab	NULL	OpenAPlaygroundWindow ExecuteTheScript Op...	Enter ESC Backspace
7	ONE FINGER CHASING TWO	g x	NULL	NULL	changeTheIPAndPort enable"ServerMode"	NULL
8	TWO FINGERS CHASING ONE	f y	NULL	NULL	SafeFile Enter Esc	NULL
9	THREE FINGERS SKIPPING TWO	Enter q Backspace	NULL	NULL	NULL	NULL

Figure 4.16: Table *overviewmenu*, stores the names[26b] of the individual *Learning Units* within the modes.



id	fiveTime	coding
1	b build.	v addAll: es. RTFlowLayout on: es. v run.
2	stream	b := RTGrapher new. b extent: 400 @ 200. RTShape withAllSubclasses
3	foo	b open. v := b view. v run.
4	[: p |	@ (RTMenuActivable new action: #inspect; item: 'browse class' action: [:e | e model browse]).

5	self	nbOfNodes := 40.
		nbOfRandomEdges := 40.
		nodes := 1 to: nbOfNodes.

6	Pharo	edges := (1 to: nbOfRandomEdges) collect: [:notUsed | nodes atRandom -> nodes atRandom].
7	true	b := RTMondrian new. b shape circle color: (Color black alpha: 0.5). b nodes: nodes.
8	false	v addAll: es. RTFlowLayout on: es. v run.

9	:=	shape := RTEllipse new
		size: #numberOfMethods;
		color: n.

10	'foo'	classes := RTObject
		withAllSubclasses.
		v := RTView new.
		v @ RTDraggableView.

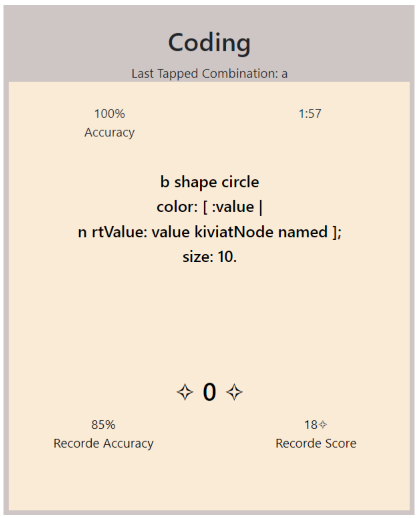
Figure 4.17: This Figure shows an extract from the table *gamesentence*.

4.3.5 Database

As mentioned in section 4.2.4, there are 18 tables in the database schema *tapstrapapp*. In the following, it is explained in more detail how the values that are written to the database are obtained. Three tables are used to store information within the *Learn2Tap* web application, which can only be changed by the administrator of the web app. These are *level*, *overviewmenu* and *gamesentence*.

Level. The *level* table (see Fig. 4.15) consists of 31 rows and stores the information, which combination results in a certain character when tapping. Each combination has an ID from the value range 1 - 31, allowing to unambiguously assign which combination results in a particular character in one of the five modes. Except for the Single-Tap mode, each mode has two columns. One column (column name without 'keycombi') contains the representation of the character, which is displayed to the user (e.g. inside the bubble in the *Level*). The column with suffix 'keycombi' stores all characters that the Tap Strap outputs when tapping the combination (explanation see 4.2.1). For Single-Tap it is not necessary to have a keycombi column, because all combinations consist of a single letter, which can be created with one keystroke.

Overviewmenu. The table *overviewmenu* (Fig. 4.16) consists of nine rows and stores the division of the *Learning Units* as well as the corresponding name of the mode, which is used in Single-Tap, Switch and Shift. The names of the individual units are taken from the Tap Strap website [26b]. The name of the mode is not used in the Double- and Triple-Tap modes, because a mode there would often contain only two characters, which would result in too many small *Learning Units* with few characters. The division of the *Learning Unit* is queried when a user opens the menu of a mode from the *Practice Part* in the user interface.



Coding

Last Tapped Combination: a

100%
Accuracy

1:57

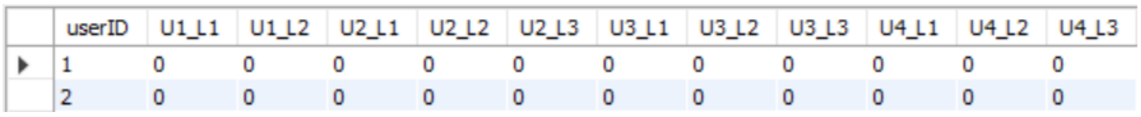
b shape circle
color: [:value |
n rtValue: value kiviatNode named];
size: 10.

✧ 0 ✧

85%
Recorde Accuracy

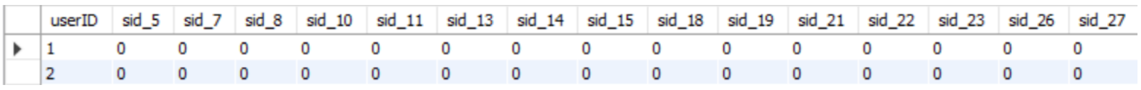
18✧
Recorde Score

Figure 4.18: Representation of the code snippet ’`b shape circle<br>color: [ :value |<br>n rtValue: value kiviatNode named ];<br>size: 10.`’ from the table *gamesentence*



	userID	U1_L1	U1_L2	U2_L1	U2_L2	U2_L3	U3_L1	U3_L2	U3_L3	U4_L1	U4_L2	U4_L3
▶	1	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0

Figure 4.19: Shows the table *tripletapstars* which stores the *Level* Reward of the Triple-Tap `Learning Units` for each user.

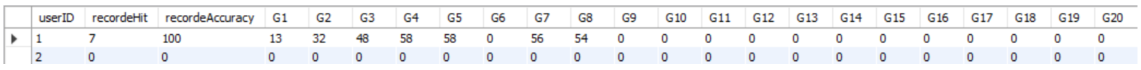


	userID	sid_5	sid_7	sid_8	sid_10	sid_11	sid_13	sid_14	sid_15	sid_18	sid_19	sid_21	sid_22	sid_23	sid_26	sid_27
▶	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.20: Shows the *tripletap* table, which stores the accuracy of the Triple-Tap mode characters for each user.

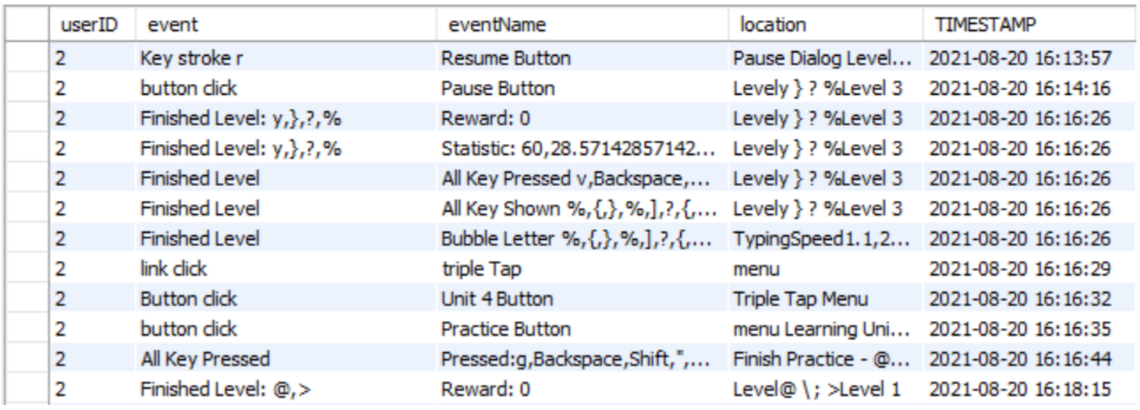
Gamesentence. This table stores 61 possible words for the game *Tap it 5-Times* and 39 possible code snippets for the game *Coding* (see excerpt 4.17). These contents are retrieved when a user opens one of the games. The code snippets contain `<br>` as a separator, so that it can be seen which part of the sentence is to be displayed in a certain line in the game when a sentence is extracted (see example Fig 4.18).

Most of the tables are used to store the learning progress of the users in the individual modes. Five tables store the reward of the *Learning Units*. The name of the table is the name of the mode with the suffix ’stars* e.g. *Singletapstars* and *shiftstars*. An example is shown in Fig. 4.19, where ’U_x’ is the unit number and ’L_x’ is the *Level* number. In the left column the user ID of the user is stored. On initialising the table, all columns except the user ID are initialised by ’0’. The maximum value of an entry is three, because a user can get a maximum of three stars per *Level*. When a user completes a *Level*, it will be compared if the new reward is higher than the old one. In this case, the new reward is entered, otherwise the old value is kept.



	userID	recordeHit	recordeAccuracy	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20
▶	1	7	100	13	32	48	58	58	0	56	54	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.21: The table shown is `timetohit` and stores the users' results for the *Time To Tap* game.



userID	event	eventName	location	TIMESTAMP
2	Key stroke r	Resume Button	Pause Dialog Level...	2021-08-20 16:13:57
2	button click	Pause Button	Levely } ? %Level 3	2021-08-20 16:14:16
2	Finished Level: y,},?,%	Reward: 0	Levely } ? %Level 3	2021-08-20 16:16:26
2	Finished Level: y,},?,%	Statistic: 60,28.57142857142...	Levely } ? %Level 3	2021-08-20 16:16:26
2	Finished Level	All Key Pressed v,Backspace,...	Levely } ? %Level 3	2021-08-20 16:16:26
2	Finished Level	All Key Shown %,{,},%,],?,{,...	Levely } ? %Level 3	2021-08-20 16:16:26
2	Finished Level	Bubble Letter %,{,},%,],?,{,...	TypingSpeed1.1,2...	2021-08-20 16:16:26
2	link click	triple Tap	menu	2021-08-20 16:16:29
2	Button click	Unit 4 Button	Triple Tap Menu	2021-08-20 16:16:32
2	button click	Practice Button	menu Learning Uni...	2021-08-20 16:16:35
2	All Key Pressed	Pressed:g,Backspace,Shift,",...	Finish Practice - @...	2021-08-20 16:16:44
2	Finished Level: @,>	Reward: 0	Level@ \ ; >Level 1	2021-08-20 16:18:15

Figure 4.22: Extract from the table *userTracking*. This table stores the users interactions with the learning application.

Another five tables store the accuracy of the characters within the modes. The table of the Single-Tap mode names the columns after the alphabet, the other tables (*doubletap*, *tripletap*, *shift*, *switch*) (see Fig. 4.20), name the columns after the character ID. Which character ID (sid_x) belongs to each character can be checked by the table *levels* (Fig: 4.15). If a new user is created, a row is added for the user, with the user ID as the primary key on the far left of the table. All characters are initialised with the default value '0'. A character can have at most the value '100' as entry, that would correspond to a 100% accuracy. The accuracy of a table is calculated as follows: $newAccuracy = (databaseAccuracy + 2 * clientAccuracy)/3$.

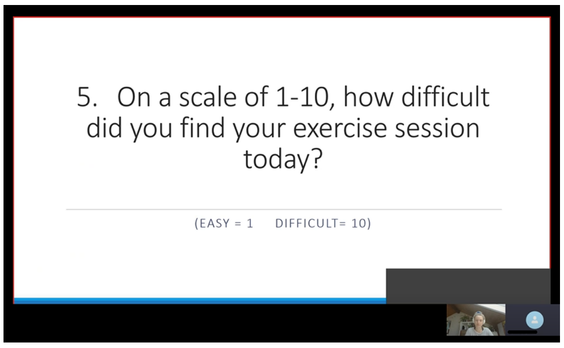
The tables *codinggame*, *fivetimes* and *timetohit* stores the results of the games (see example Fig. 4.21). The record accuracy or the record score will be overwritten if a higher value is reached in the corresponding record. The general score (G1 - G20) is calculated as follows: $G1 = (score + accuracy)/2$. During the creation of a user, all columns except 'userID' are initialized with '0'.

In the table *user* an ID is stored, which is the primary key and is assigned incrementally. Furthermore, a user name is stored, these are also unique in the table. This ensures a function within the server. In the column 'hand', depending on the selected hand, the value is 'left' or 'right'.

The table *usertracking* (see excerpt Fig. 4.22) stored the activities of the users. What kind of user events exist, can be taken from 4.2.3. The column 'userID' contains the ID of the user who performed the action. The columns 'event', 'eventName' and 'location' store as text the data which are transmitted by the front-end. The column 'timestamp' contains the time at which a row was created in the table.

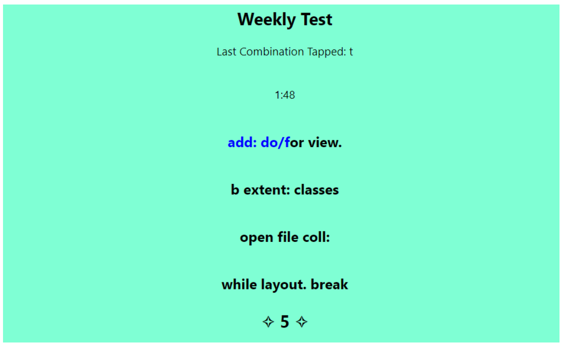
5 Evaluation

To understand the impact of *Learn2Tap*, a real-life long-term case study was conducted. This chapter includes the design, the results and a discussion part of the study. The part about the design of the study contains the subsections: Protocol, Data set, Techniques, Participants, Apparatus and Data collection. The object of the study to be investigated is the learning application *Learn2Tap*. The aim is to observe the use of *Learn2Tap* by the participants. To be able to assess the qualities in terms of effectiveness and usability. The study is described from the participant (end-user) point of view and is a long-term study in the context of the project "On-the-go Authoring Visualisations through Wearable Keyboards".


5. On a scale of 1-10, how difficult did you find your exercise session today?

(EASY = 1 DIFFICULT= 10)

(a) Interview via Skype.


Weekly Test

Last Combination Tapped: t

1:48

add: do/for view.

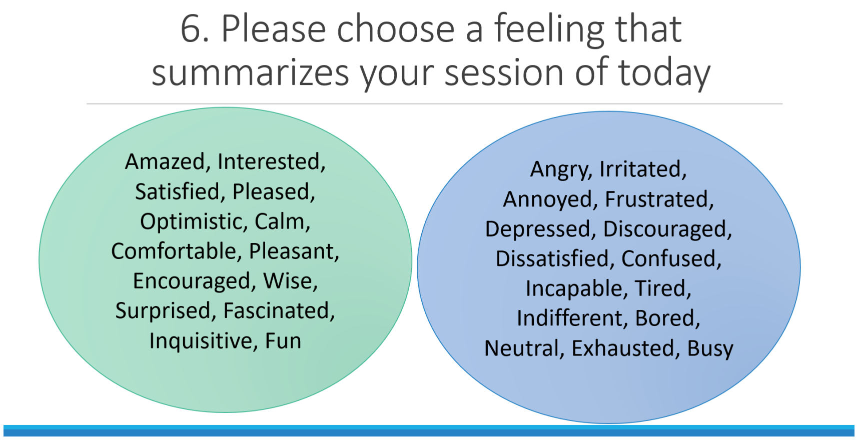
b extent: classes

open file coll:

while layout. break

✧ 5 ✧

(b) Possible Weekly Test.


6. Please choose a feeling that summarizes your session of today

Amazed, Interested, Satisfied, Pleased, Optimistic, Calm, Comfortable, Pleasant, Encouraged, Wise, Surprised, Fascinated, Inquisitive, Fun

Angry, Irritated, Annoyed, Frustrated, Depressed, Discouraged, Dissatisfied, Confused, Incapable, Tired, Indifferent, Bored, Neutral, Exhausted, Busy

(c) Feelings to be selected.

Figure 5.1: Figure (a) shows the interview about Skype; (b) is the third weekly test; (c) is the slide with the feeling to be selected for daily sessions.

5.1 Design

The study design is a quasi-experiment which is carried out over a longer period of four weeks and follows the “Remote Synchronous Usability Study” strategy [CG11]. The study is conducted via Skype[1]. There are two types of sessions: the *Daily Sessions* which last 20 minutes, in which the participant has 15 minutes to use *Learn2Tap* and afterwards the participant is asked the same questions. Once a week there is a *Weekly Session*, which starts with the *Daily Session*, followed by more questions and a tap test to assess the user’s performance. The questions that occur in the session are conducted in the form of interviews, in which the participant is asked questions by the study leader. To collect subjective impressions of the learning application. The users activities in *Learn2Tap* are stored in the database and constitute the objective data of the study.

5.1.1 Protocol

All sessions are conducted via Skype and start with the study participant calling, i.e.. the study participant is called by the study leader. In total, there were four different types of sessions. The *First Session* and *Last Session* were conducted once. The *Daily Session* was conducted a total of 20 times and was also a component of the other 3 session types. The *Weekly Session* was conducted 3 times and another time as part of the *Last Session*. All interviews conducted during the sessions were recorded via Skype. The participants turned off the camera during the interview to ensure their anonymity. The current question being discussed during an interview is shown at that time on the slide of a PowerPoint presentation (Fig. 5.1a).

First Session. Before the study can be started, *Learn2Tap* and MySql must be installed on the participant’s computer. Furthermore, the Tap Strap 2 must be configured with the mapping for *Learn2Tap*. For this purpose, the user receives a *Preparation Sheet* (A.4.3) which contains the information for the configuration. In the *First Session*, the context of the user study is explained to the participant. Before the study starts, the participant fills in the *Consent Agreement* (A.4.1) and *Privacy Information* (A.4.2), with which the user agrees to participate in the study and allows the use of the data. The *Demographic Questionnaire* (A.4.4) is used to record demographic data and is to be completed by the participants. After receiving the forms, the participant is shown a short introduction video about *Learn2Tap*, which explains the interface of the application. Afterwards, the user receives the document *Order Learning Units* (A.4.5), which specifies the order in which the modes are to be learned, as well as the point in time at which certain games can be played. This concludes the preparations for conducting the study and the participant is given 15 minutes to use *Learn2Tap*, after which they are contacted again by the study leader. The user sends the study leader the status of the database. At the end of the session, the participant will be asked the *Daily Questionnaire* and the questionnaire that will record the first impression of the learning application (see A.5.1 for *First Session Questionnaire*).

Daily Session. The *Daily Sessions* starts with the users resetting the table *userTracking* using the ’Truncate’ function and starting the *Learn2Tap* application. Afterwards, the participant is given 15 minutes to use *Learn2Tap* independently. At the end of the 15 minutes, the study leader contacts the

[1]https://www.skype.com/de/, accessed 06.09.2021

participant again. The participant stops the application and sends the study leader the status of the database. Finally, the participant is asked the questions of the *Daily Questionnaire*. The questions of the *Daily Questionnaire* are listed below:

1. How well were you able to concentrate?
2. What did you learn?
3. What difficulties did you encounter?
4. What new things did you discover today that excited/annoyed you?
5. Do you think you will be able to program in VR by the end of the 4th week?
6. On a scale of 1-10, how difficult did you find your exercise session today? (1 = easy, 10 = difficult)
7. Please choose a feeling that summarizes your session of today. (See Fig. 5.1c for possible feelings.)

Weekly Session. The *Weekly Session* starts with the *Daily Sessions*. Following the *Daily Sessions*, a test is conducted. The test lasts two minutes and contains a text, based on the characters the user has learned so far. This text to be tapped. The test can be used to record the user's progress. For example, the test may look like Fig. 5.1b. After completing the test, the participant sends a test text file containing the result of the test to the study leader. Finally, the user is asked some interview questions (see A.5.2, A.5.3, A.5.4 for *Weekly Questionnaires*) that specifically target problems or abnormalities of the last study week, an example questions is: "Do you think 15 min is too short to play a game and continue learning combinations?".

Last Session. The *Last Session* begins with the performance of the *Daily Sessions*. Following this, the user receives their fourth weekly test. In addition, the participant repeats their previous three weekly tests to compare their learning progress. Finally, a final interview (A.5.5) is conducted to determine the participant's impression of *Learn2Tap*, as well as the users experience with the Tap Strap.

5.1.2 Data set

The application uses data read from the database with the help of the back-end (for more details see section 4.3.2). The database stores which Tap Strap combination puts out a particular character (see Fig. 4.15). Depending on the frequency with which a character is used in AVAR, the corresponding combination is determined. The more often a character is used, the easier the Tap Strap combination (see section 3.1). In addition to the usual characters available on the keyboard, the Tap Strap has the Shift mode. This allows the user to use shortcuts that can be used later in AVAR like 'CreateTestMethodeAndJumpToIt' or 'OpenAPlaygroundWindow'. In the game part of *Learn2Tap*, words and code snippets are read from the database. These code snippets are written in Pharo and Rossal2.

5.1.3 Technique

The aim of *Learn2Tap* is to help a user learn the tap combinations of the mapping for AVAR 3.2 of the Tap Strap. The application contains four *Main Menu Items* that are used frequently and two other items that are used less regularly. The four main items are *Glossary*, *Practice*, *Games* and *Statistics*. *Glossary* shows the mapping of the Tap Strap combination to the different characters. In the *Practice Part*, which is subdivided according to the different modes, the various combinations can be practised. *Games* serve to consolidate the combinations learned and prepare the user to write code. In the *Statistics* section, users can find their statistics on the accuracy of the individual characters and the points achieved within a game. The two other menu items are *User Administration* and *Hints*. The menu option *User Administration* enables the user to create a new user or to select an already existing user. Under the menu option *Hints*, the user can access helpful tips to facilitate the use of the learning application. Details can be found in chapter 4.

5.1.4 Participants

Both study participants (24 and 26 years old) are female, currently studying for a master's degree in computer science and have a bachelor's degree in computer science. They have no experience with the Tap Strap or any other chorded keyboard. Recruitment was based on the fact that they both have a job as student assistants at the Visualisation Research Center of the University of Stuttgart.

5.1.5 Apparatus

The devices used are the Tap Strap 2 (see also 3.1) and the laptops of the study participants (ThinkPad P14s Gen.1 and HP Notebook - 15s-dr0002tu). The laptops have Window 10 as operating system, which is relevant for the installation of *Learn2Tap*. The Tap Strap is an input device from the category of wearable keyboards. It consists of five flexible rings that are pulled over the fingers. There are three different modes in total: Optical Mouse, Air Gesture and Tapping. The default mode is the Tapping mode and the only relevant mode for the learning application. Tap Strap is used with a customised mapping, which is suitable for programming Pharo (see also section 4.2.1).

5.1.6 Data collection

The data collection will use a *Demographic Questionnaire*, Interviews, Daily Learning Status, Weekly Tests and User Tracking.

Demographic data. The *Demographic Questionnaire* (A.4.4) is used to record the demographic data of the study participants, including age, gender, education level and whether they have had previous experience with the Tap Strap or another chorded keyboard.

Interviews. In the interviews, the participant is asked open questions and the experiences with the *Learn2Tap* application are shared. Of particular interest are problems, suggestions for improvement and positive aspects of the learning application. During the study, a total of four different interviews will be conducted. One is the *Daily Session* interview, the questions of which are described in 5.1.1. Then the *First Session* (A.5.1) interview, to capture the participants first impression of *Learn2Tap*.

Then there are three *Weekly Questionnaire*(A.5.2, A.5.3, A.5.4) interviews to clarify uncertainties and open questions that arose during the week. The fourth interview is the *Final Questionnaire* interview (A.5.5), which is intended to determine the participants overall impression of *Learn2Tap*. All interviews will be recorded using the Skype Recorder function. The participants has turned off the camera to preserve their anonymity. The questions are asked using a Power Point presentation, with one question on each slide. After the session, the participants answers are transcribed.

Daily learning status. As explained in section 4.2.4, there are 13 tables that are used to store a user's learning status. At the end of each session, the study participant sends the status of these tables to the study leader so that it can be tracked what progress a user has made each day. For this purpose, a database schema was created for each study participant containing the tables that store the learning progress (e.g.. accuracy of individual characters, reward received in the modes). Instead of the 'userID', the day of the study was entered so that it can be clearly assigned which data belong to which day.

Weekly test. The weekly test (see example Fig. 5.1b) contains characters that the participant has already learned. The content is based on programming in AVAR. The user is given two minutes to type as much of the text as they can manage. If the participant does not know a character, they can skip it by pressing the right arrow key. The test measures the following data: Typing speed, words per minute, accuracy and error rate. In total, there are four different weekly tests, whereby tests 1 - 3 are repeated in the *Last Session* to compare whether the user has improved or worsened their tap skills.

User tracking. User tracking is used to store detailed information about the participant's interactions with *Learn2Tap*, for details see section 4.2.3. These are stored in the table *userTracking* and for each interaction the 'userID', name of the object with which the interaction is carried out, location of the interaction and the timestamp at which the interaction was carried out (Fig. 4.22, shows excerpt from the database). The *userTracking* sent at the end of a *Level* or *Game* is of particular interest, as this also contains information about the typing speed and which characters were tapped before the correct one was entered.

5.2 Results and Discussion

This chapter serves to presents the results of the long-term study. First, the results of the *Daily Sessions* are presented, divided into the learning modes, which are Single-Tap, Double-Tap, Triple-Tap and Switch. The accuracy, error rate and typing speed of the different modes are discussed, as wells as the *Difficulty Scale*. Afterwards, the results of the weekly tests are presented and analysed. Particularly interesting are the accuracy, the error rate, the typing speed and the words per minute. Finally, the chapter is concluded with a summary of the results.

5.2.1 Recorded data

In this section, the quantitative data collected within the *Learn2Tap* application is presented and analysed. The data is taken from the database that belongs to *Learn2Tap*. Mainly the accuracy and typing speed of the individual characters and the average resulting from this are analysed.

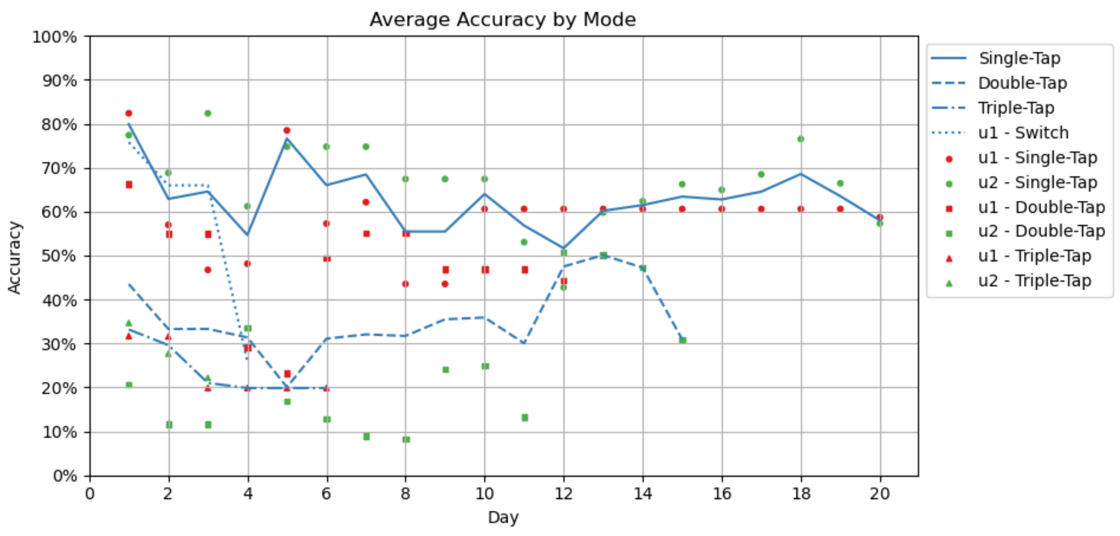


Figure 5.2: The graph shows the progression of the accuracy of each mode. The number of the day refers to the number of days how long the mode has been practiced. The points are the exact values of the individual participants.

Single-Tap

The Single-Tap mode was the first mode learned by the study participants. The first participant (u1) finished learning the mode on day 8, the second participant (u2) on day 5.

Accuracy. The accuracy of the Single-Tap mode was around 58% (Fig. 5.2) on the last day of the study. U1 finished learning the Single-Tap mode on day 8, with an accuracy of about 45%. U2 finished learning the Single-Tap mode on day 5 with an accuracy of 75%. The change in accuracy by the end of the user study is due to re-practicing *Learning Units* of Single-Tap mode or playing *Time To Tap*. Fig. 5.4 shows the accuracy of the Single-Tap characters with their corresponding tap combination. Once the accuracy is shown on the day of finishing the learning of the Single-Tap mode and the other time on day 20. When we look at Fig. 5.5, it is noticeable that both participants achieved their highest accuracy on the first day. This is since the number of fingers increases with the progress in the *Learning Units*, i.e.. on the first day the easy to perform combinations were learned that use one finger or two fingers that are next to each other. We identify that the accuracy of u1 drops significantly on days 1 too 3, as well as on day 6. The accuracy of u2 drops for the first time on day 2. For both participants, the drop in accuracy is due to the fact that they accidentally switch to switch mode when typing the letters ’o’, ’p’, ’s’ or ’r’. Since the combination of these letters differs only by a finger tap from the combination that is used to switch to switch mode. This is also shown in Fig. 5.4, because among the worst eight characters when finishing the Single-Tap mode are ’p’ and ’r’. When we look at the diagrams of the accuracy of ’s’, ’r’ and ’p’ (Fig. 5.6) we notice that at the beginning (before exiting the Single-Tap mode) their accuracy is more often below 50%. Another point to note is that u1’s scores are relatively stable from day 10 onwards, which is since the participant ended the Single-Tap mode on day 10 and then only started playing *Games* towards the end. And these were then also predominantly played in Double- and Triple-Tap mode. At the end of the 20 days, ’s’ and ’r’ (see Fig. 5.4) are among the best performers, as the users type more accurately over the course of the study. Looking again at the accuracy of u1, it is

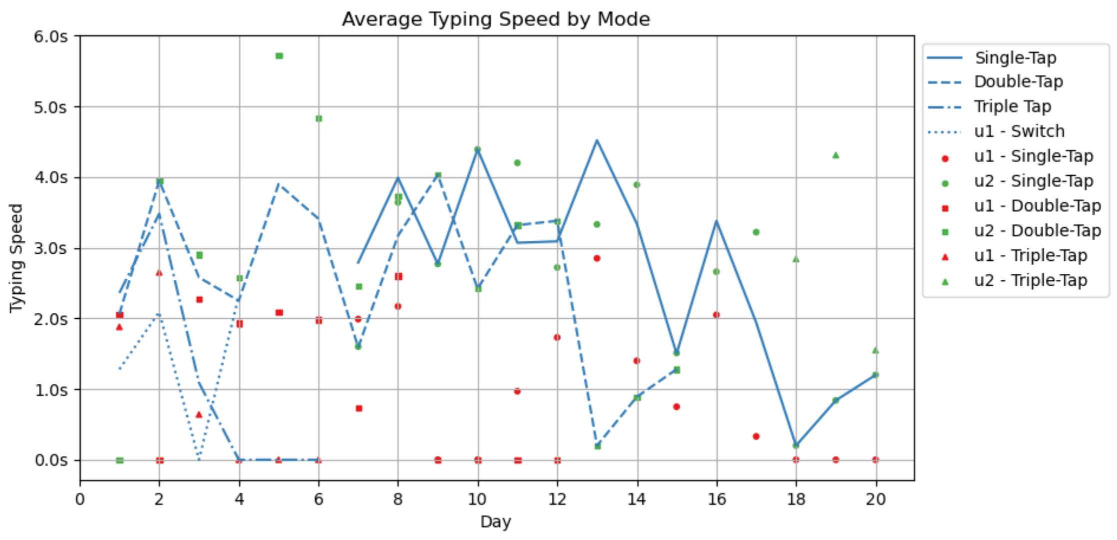


Figure 5.3: The graph shows the evolution of the typing speed of the individual modes. The number of the day refers to the number of days how long the mode has been practiced. The line of the Single-Tap mode starts on day 7, because the typing speed was measured from day 7 of the study. The points are the exact values of the participants.

noticeable that the accuracy drops noticeably again on day 7. The participant stated that she had difficulty remembering the combination 'f', 'y' and 'g'. This is also reflected in the diagram of Fig. 5.4, as none of the letters had an accuracy higher than 55 % when finishing the learning of the Single-Tap mode. For u2, the accuracy increases briefly from day 5, as the participant noticed that her keyboard was not set to US keyboard. As a result, when she tapped 'z', a 'y' always appeared.

Another striking feature is that the accuracy values from Fig. 5.4 change from the day the Single-Tap mode is ended until Day 20. This is because at the beginning of the *Learning Unit*, the aim was to learn the characters and therefore their tap combination. Low accuracy then comes from the user having problems typing or remembering a combination accurately. After finishing the Single-Tap mode, the values mainly change by playing *Time To Tap*, in which the user has one try to tap the character correctly. In particular, the characters of the last two *Learning Units* ('f', 'y', 'Enter', 'q', 'Backspace') performed poorly in the game. This could be because these characters were less repeated later, as they were learned in the last two sessions.

It is interesting to note that the accuracy of 'e' (Fig. 5.6a) and 't' (Fig. 5.6b) varies greatly, even though they have a simple combination that should be easy for the user to remember. Furthermore, tapping the combination should not be a problem. These two combinations explain why even in Single-Tap mode the accuracy is not above 60%. As both participants said in the interviews, it is difficult for them to remember the combinations permanently, because there are a lot of them. On day 5 of the user study, both participants mentioned for the first time that they had problems remembering the new combinations. When we look at the lines from u2 of the diagrams in Fig. 5.6, we notice that the lines fluctuate strongly after learning the Single-Tap mode. The drop in the accuracy shows that the participant has problems remembering the combinations permanently. It is hard to determine when users started having problems remembering the characters they had already learned, as many difficulties arose during the study that affected accuracy.

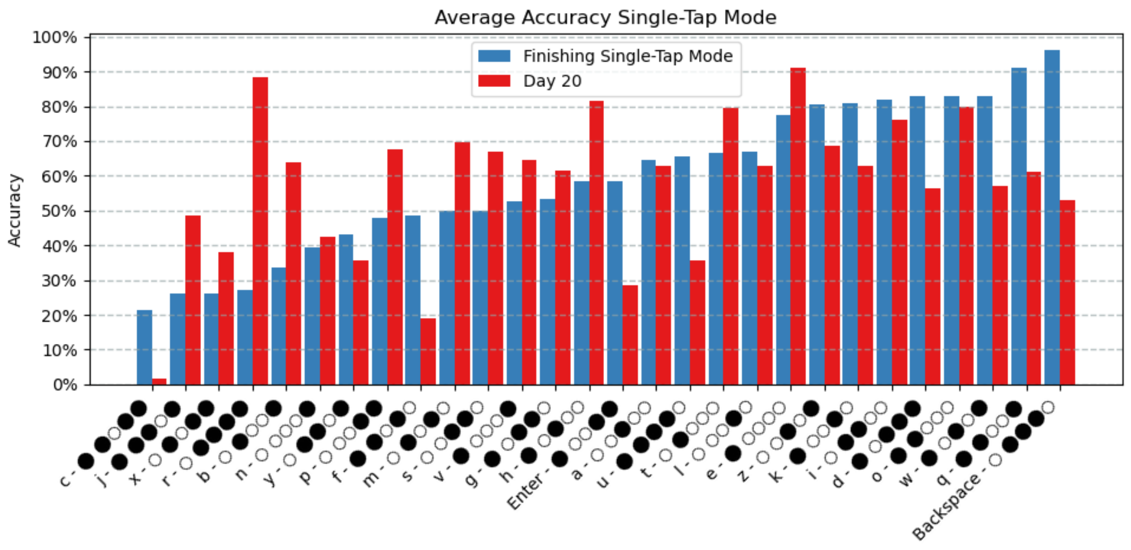


Figure 5.4: The above diagram shows the accuracy values of the individual characters in Single-Tap mode. Once for the day on which the participants had fully learned the Single-Tap mode and secondly for the last day of the study (day 20).

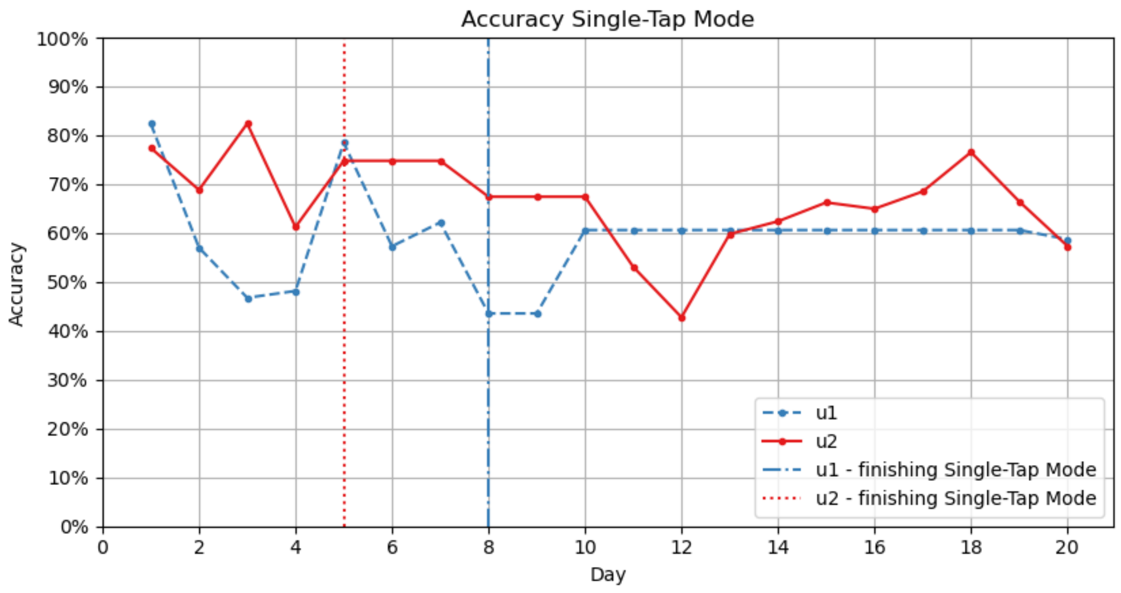
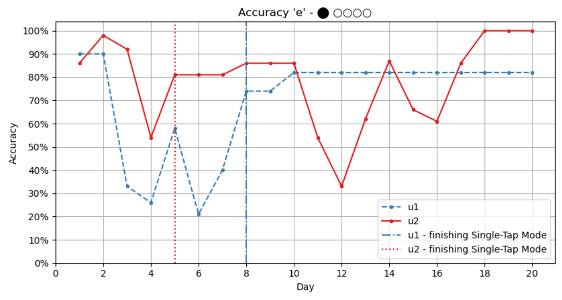
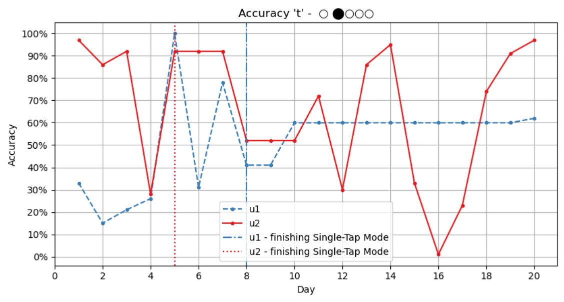


Figure 5.5: The above diagram shows the accuracy value progression of the Single-Tap mode. The vertical lines at day 5 and 8 represent the end of the learning of the Single-Tap mode.

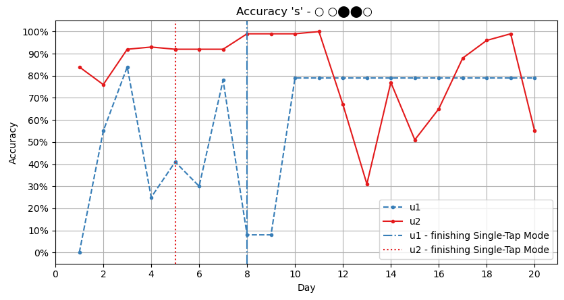
Error rate. The error rate is calculated as follows: $ErroRate = 100 - Accuracy$. The error rate in Single-Tap mode on day 20 is about 42%. When we look at Fig. 5.4, we notice that difficult combinations such as the 'c' have a low accuracy on day 20, which results in a high error rate. Furthermore, characters that are used less frequently (compare statistics Pharo, see appendix A.1) and learned at the end of the mode ('Enter', 'y', 'x') have a higher error rate.



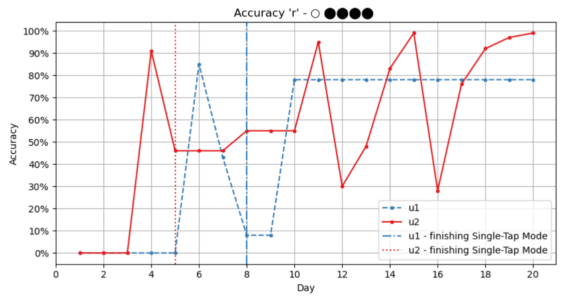
(a) Accuracy of ’e’.



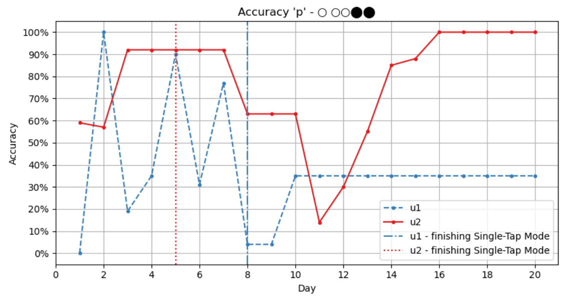
(b) Accuracy of ’t’.



(c) Accuracy of ’s’.



(d) Accuracy of ’r’.



(e) Accuracy of ’p’.

Figure 5.6: The above diagrams show the accuracy of letters in Single-Tap mode. ’e’ and ’t’ are selected because they have a simple tap combination. ’s’, ’r’ and ’p’ are shown because they have combinations that allow the user to accidentally switch modes when tapping inaccurately.

Typing speed. The average typing speed is 1.55 seconds for u1 and 2.55 seconds for u2, this results in an average value of 2,05 seconds (see Fig. 5.3). Typing speed was only measured from day 7 onwards. The diagram of Fig. 5.7 shows the average typing speed of the characters from Single-Tap mode. Here ’g’ with 1.82 seconds represents the lower limit of the typing speed, and ’q’ with 4.04 seconds the upper limit. Surprisingly, ’q’ has the highest typing speed and at the same time the second highest accuracy value on the day the participants had finished learning the Single-Tap mode. On day 20, the accuracy of ’p’ lost about 30%. The long typing speed and decreasing accuracy could be caused because the users had problems remembering the combination towards the end of the study. Furthermore, it is noticeable that combinations with fewer fingers have a better typing speed. A possible explanation is that the fingers can be positioned faster for the corresponding tap combination.

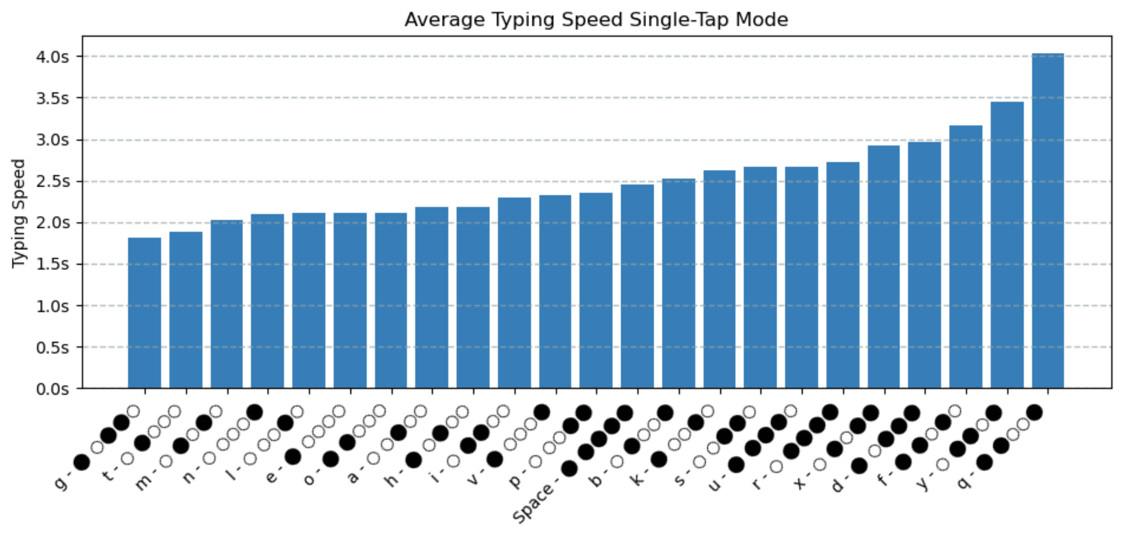


Figure 5.7: The above diagram shows the typing speed of the individual characters in Single-Tap mode.

One reason for the low typing speed is that the participants must type the combinations precisely and think about what the corresponding tap combination is before tapping. Taking the average typing speed for a letter in Single-Tap mode and using this, to calculate the words per minute [ZHS02], the result is 7.74 wpm for u1 and 4.7 wpm for u2. Comparing this value to the wpm of the fixed keyboard, the Tap Strap performs poorly in the simplest tap mode.

Summary: Both participants achieved the highest accuracy on the first day, accidentally switched to switch mode, and had problems remembering the combinations for the first time on day 5. Another problem was that one participant's keyboard layout was not set to US keyboard layout. It is interesting to note that combinations learned in the last two *Learning Units* have a lower accuracy because they are repeated less frequently. Also combinations that are difficult (e.g. 'c') have a lower accuracy. Overall, the precision of tapping increases over the course of the study.

Double-Tap

Accuracy. The average accuracy over the duration of the study was approximately 46% in Double-Tap mode. On the last day (see Fig. 5.2) the accuracy was around 38%. U1 finished learning the Double-Tap mode with an accuracy of about 54% on day 14 and needed 7 days in total. U2 achieved an accuracy of about 51% and finished the mode on day 17 after 11 days. Fig. 5.8 shows the accuracy of the individual characters of the Double-Tap mode, once on the day of finishing learning the Double-Tap mode and once on day 20.

When looking at the course of accuracy (Fig. 5.9), it is noticeable that this was very low in the first period of learning, e.g.. day 5 - u1 = 25% or day 8 - u2 = 9%. First, we look at the course of u1, which drops for the first time on day 2. The *Learning Unit* ': . /' was practised, which

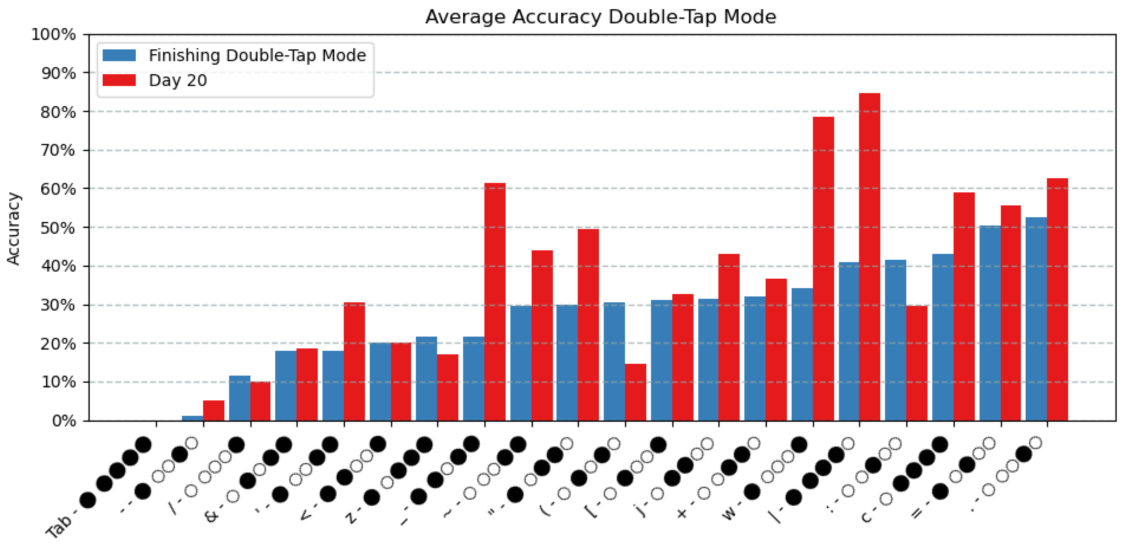


Figure 5.8: The above diagram shows the accuracy values of the individual characters in Double-Tap mode. Once for the day on which the participants had fully learned the Double-Tap mode and secondly for the last day of the study (day 20).

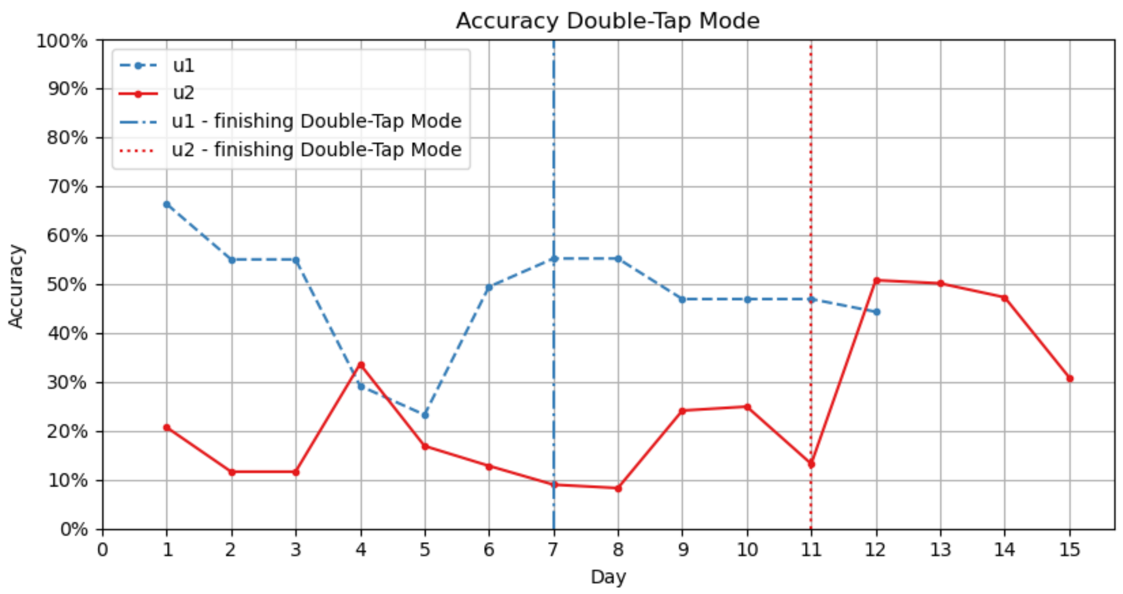


Figure 5.9: The above diagram shows the accuracy value progression of the Double-Tap mode. The vertical lines at day 7 and 11 represent the end of the learning of the Double-Tap mode.

have the combinations 2× ○ ○ ● ○ ○, 2× ○ ○ ○ ● ○, 2× ○ ○ ○ ○ ●. During thedouble taps of these combinations, the student switched the uppercase mode more often (2× ○ ● ○ ○ ○). Accuracy did not change from day 2 to 3 because the participant could not successfully complete any *Learning Unit*. The reason for this was that 'Cap lock' was activated on the computer keyboard, and therefore the tap combinations were not correct. The decrease in accuracy on day 4 is because the characters '_' and '-' within the bubbles were difficult to distinguish (Fig. 5.8). On day 5 the application crashed several times so that the participant could not successfully complete a *Learning Unit*. Secondly, the accuracy of u2 within Double-Tap is considered. The first problem occurred on day 2, when she had difficulty performing the double tapping correctly because the speed of the tapping had to be correct. Another problem (day 5) was that when the 'z' was tapped, she switched to switch mode and the second tap changed the mapping. On day 11, the main problem was that when 'Tab' was tapped, a button within *Learn2Tap* was marked and pressed with the next tap. This is also the reason for the low accuracy (Fig. 5.8) of 'Tab'. If you look at (Fig. 5.8) more closely, you will notice that '/' belongs to the worst values, although the combination seems to be simple. A good Double-Tap combination seems to be the 'c' (Fig. 5.8). Overall, it is striking that there are hardly any values with an accuracy above 80%.

Error rate. The error rate in Double-Tap mode is significantly higher than in Single-Tap mode, at around 53,5% . The most common problems with the Double-Tap mode were switching to the wrong mode with changing the mapping, accurately executing the Double-Tap with the corresponding finger, as well as with *Learn2Tap* (crashed a few times) and remembering the different combinations.

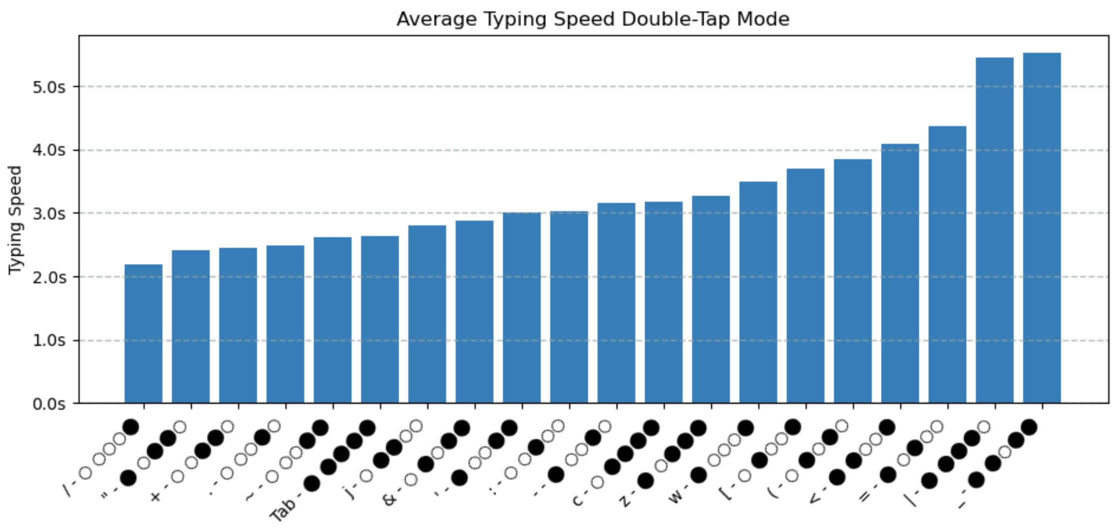


Figure 5.10: The above diagram shows the typing speed of the individual characters in Double-Tap mode.

Typing speed. The typing speed of the Double-Tap mode is higher than that of the Single-Tap, with approximately 2.75 seconds (Fig. 5.3). When looking at Fig. 5.10, we can seen that the fastest typing speed is at '/' with 2.2 seconds. The upper limit is 5.5 seconds ('|') and 5.6 seconds ('_'). The reason for the long typing time of '|' could be that it was accidentally switched to shift mode (combination to change to switch mode ○ ○ ● ● ●). One of the reasons for the bad typing speed of '_' is because u1 had problems distinguishing ' _ ' and ' - ' as mentioned in the previous section.

In Double-Tap mode, the words per minute is 6.3 wpm for u1 and 4.0 wpm for u2.

Summary. The problems were switching into the uppercase mode, as well as switching into the switch mode and then changing the mapping. The participants had problems at the beginning to hit the speed of the Double-Tap. On one day, one participant had 'Cap lock' activated on the computer keyboard, so that the tap combinations produced the wrong characters. In addition, *Learn2Tap* chrashed a few times and the characters ’-’ and ’_’ were difficult to distinguish within the bubbles. When performing the combination for ’Tab’, a button was marked first which was clicked with the next tap, therefore ’Tab’ has an accuracy of 0%. Overall, it is noticeable that there are few accuracy values above 80%.

Triple-Tap

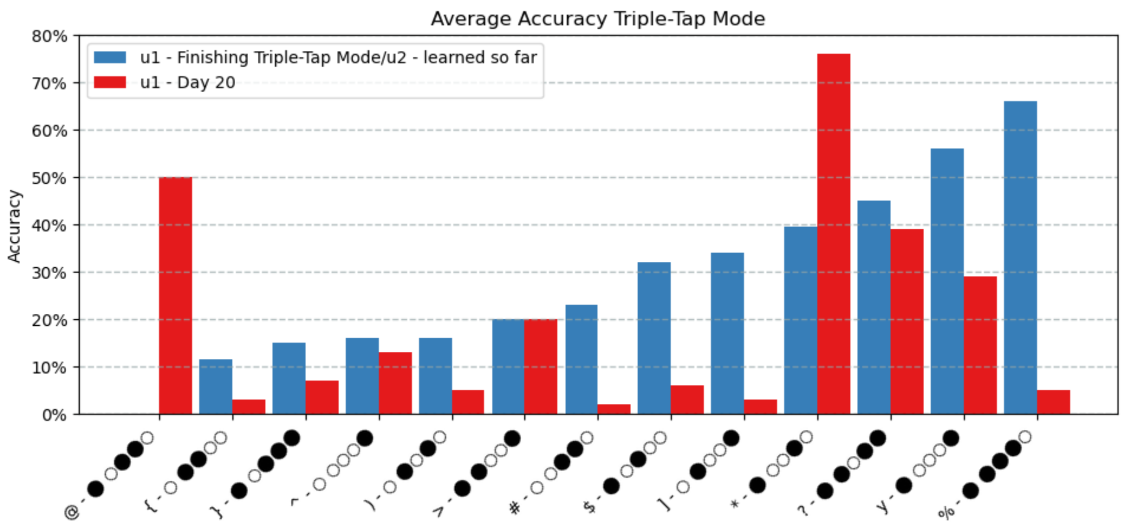


Figure 5.11: The above diagram shows the accuracy values of the individual characters in Triple-Tap mode. Once for the day when u1 had fully learned the Triple-Tap mode, offset by the accuracy of u2’s characters learned so far. As well as the accuracy for u1 on the final day of the study (day 20).

Accuracy. The average accuracy of the Triple-Tap mode is 21.25% (Fig. 5.2). U1 completed the Triple-Tap mode on day 16 after 2 days of practice. U2 has learned the first 6 characters of the mode by day 20. Fig. 5.11 shows the accuracy of the individual characters in Triple-Tap mode. It is striking that the highest achieved accuracy value is 76% for the character ’*’. Furthermore, it is noticeable that more than half of the characters in Triple-Tap mode have an accuracy of less than 10%, on day 20 of the user study. According to the interviews, the reason for this low accuracy is that it is more difficult to remember the combinations and to match the typing speed of the Triple-Tap. U2 also has the problem of switching to shift mode more often by accident.

Error rate. The error rate of the Triple-Tap mode is 78.75%.

Typing speed. The average typing speed is 2.3 seconds (see Fig. 5.3), which is better than the typing speed in Double-Tap mode. Fig. 5.13 shows the typing speed of the individual characters in Triple-Tap mode. The lower limit is 2.25 seconds and the upper limit is 3.6 seconds. There is no apparent pattern as to why certain combinations are tapped faster and others slower.

The words per minute for the Triple-Tap mode is 7.1 wpm for u1 and 4.1 wpm for u2.

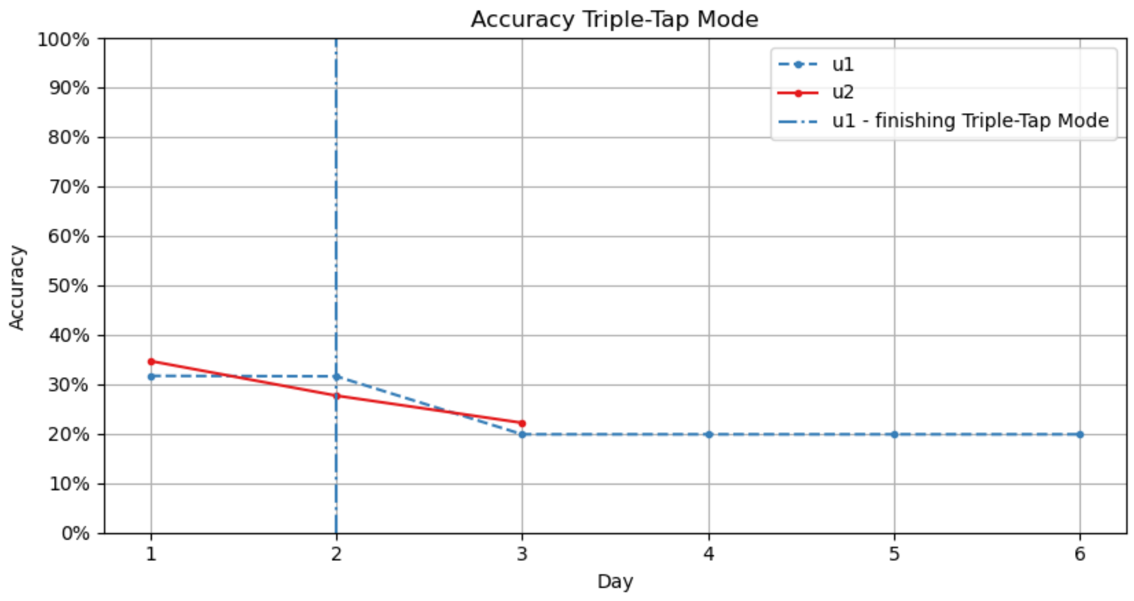


Figure 5.12: The above diagram shows the accuracy value progression of the Triple-Tap mode. The vertical line at day 2 represent the end of the learning of the Triple-Tap mode from u1.

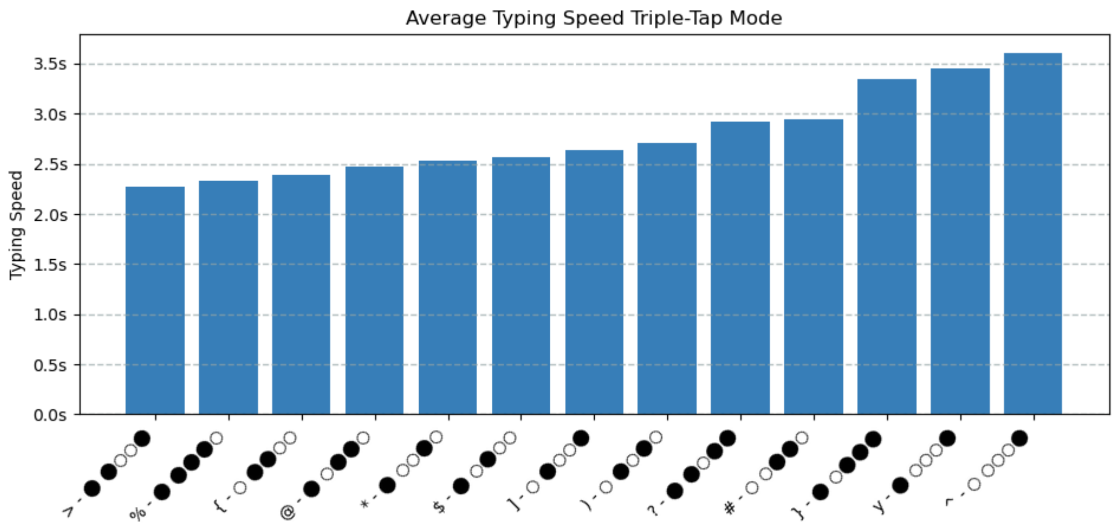


Figure 5.13: The above diagram shows the typing speed of the individual characters in Triple-Tap mode.

Summary. The problems with the Triple-Tap mode were that the participants had difficulty hitting the speed of the Triple-Tap, remembering the combinations, and switching to Shift mode. It is striking that on day 20 of the user study, more than half of the characters from the Triple-Tap mode had an accuracy below 10%.

Switch

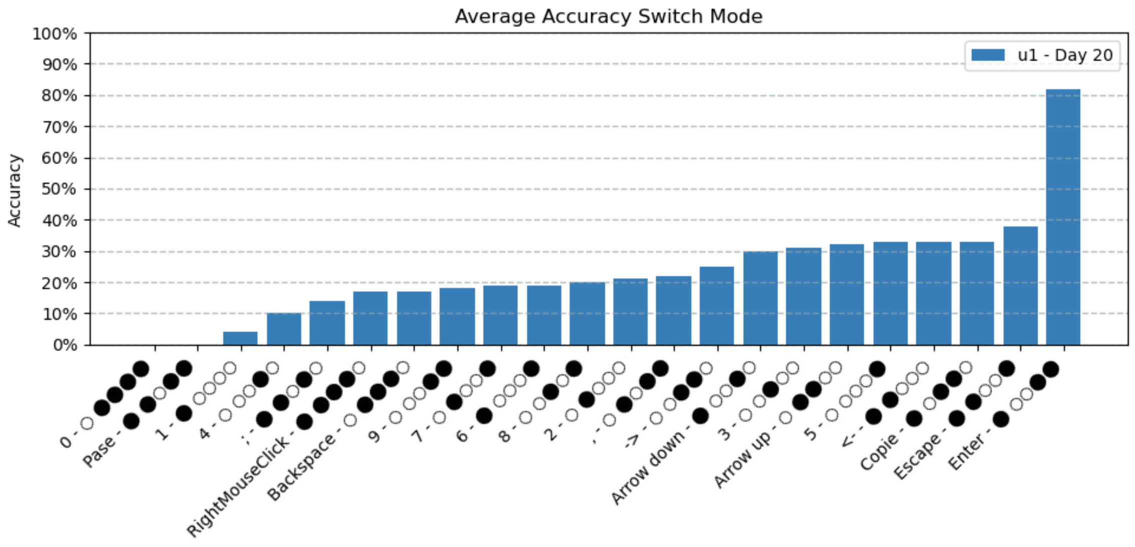


Figure 5.14: The above diagram shows the accuracy value of the individual characters in Switch mode.

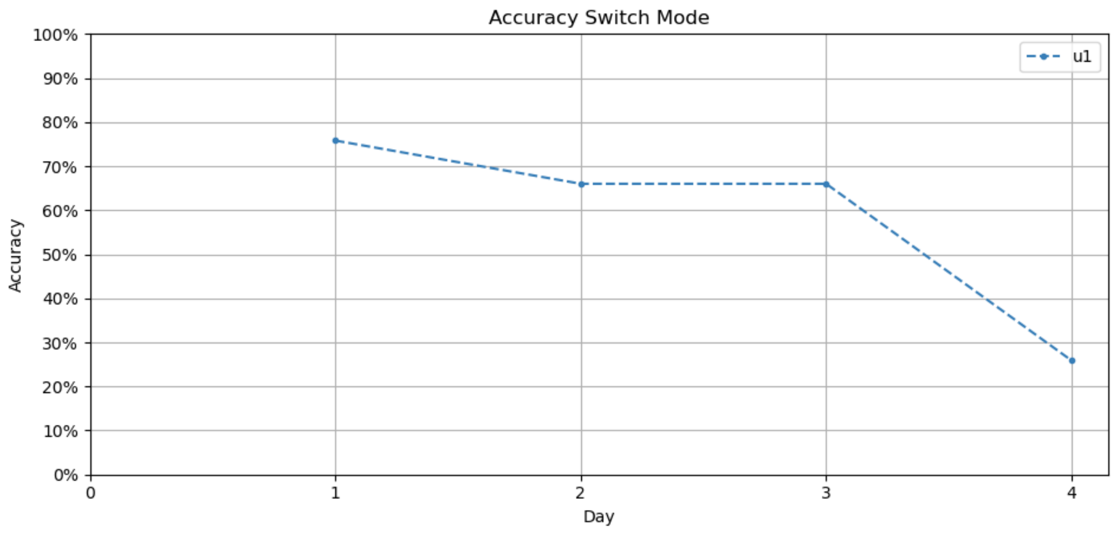


Figure 5.15: The above diagram shows the accuracy value progression of the Switch mode. U1 finishes learning the switch mode after 4 days.

Accuracy. Study participant u1 completed the Switch mode on day 20 with an accuracy of 25% (Fig. 5.2). Fig. 5.14 shows the accuracy of the Switch mode characters, as well as the corresponding combination. On the first day of learning the Switch mode, the accuracy was 80% (Fig. 5.15) and decreased continuously thereafter. The drop of about 10% in accuracy on day 2 is caused by the fact that the participant could not distinguish between the 'ArrowUp' and 'Arrow Down' characters. The stagnation of the accuracy on day 3 was related to the difficulty of the participant to successfully complete a *Learning Unit*, because the Tap Strap did not tap any characters despite the correct combination. A possible cause for the problem could be a low battery level. The status of the battery does not correspond with the battery indication on the computer (according to the computer, her Tap Strap still had 70% battery). Another problem on that day was that the Tap Strap did not hit the combination of 'copy', this was because the wrong key combination was stored in the database. The problem was solved immediately by storing the correct key combination in the database. A decrease in accuracy on day 4 was caused by the problem that the Tap Strap still did not output any characters. Considering the Fig. 5.14, it is noticeable that only the character 'Enter' has a high accuracy (82%). The remaining characters have an accuracy of less than 40%. Despite the low accuracy, u1 stated that the combinations of the switch mode were easy to learn.

Error rate. The error rate of the Switch mode was 85% and can be attributed to the 'non-functioning' of the Tap Strap on the last day.

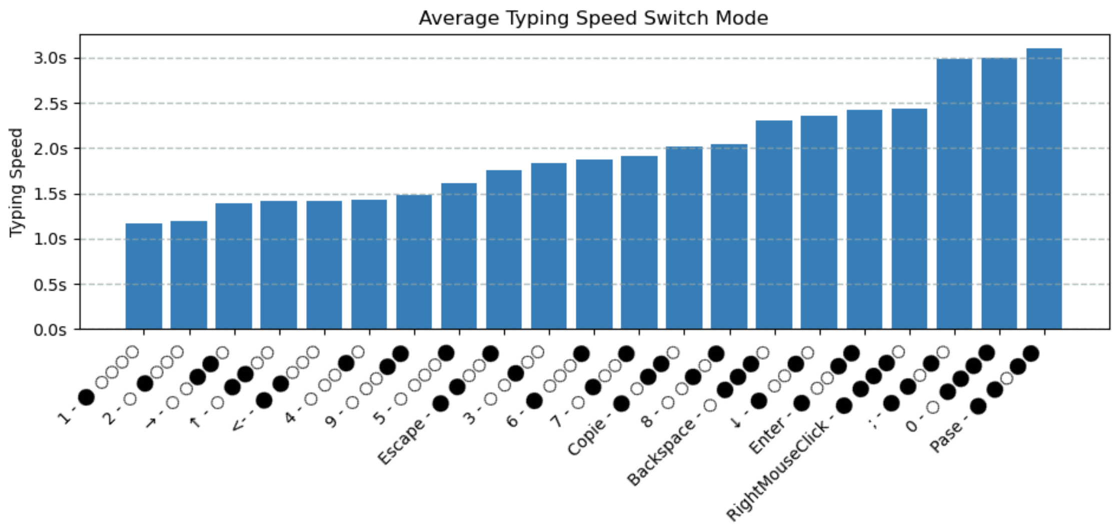
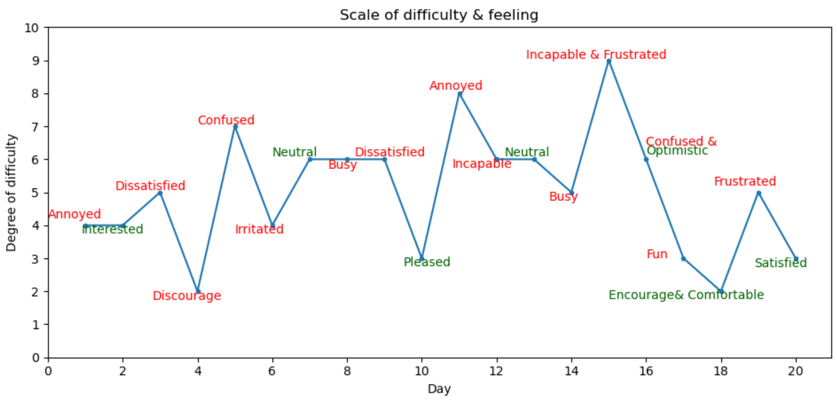


Figure 5.16: The diagram shows the typing speed of the individual characters in Switch mode.

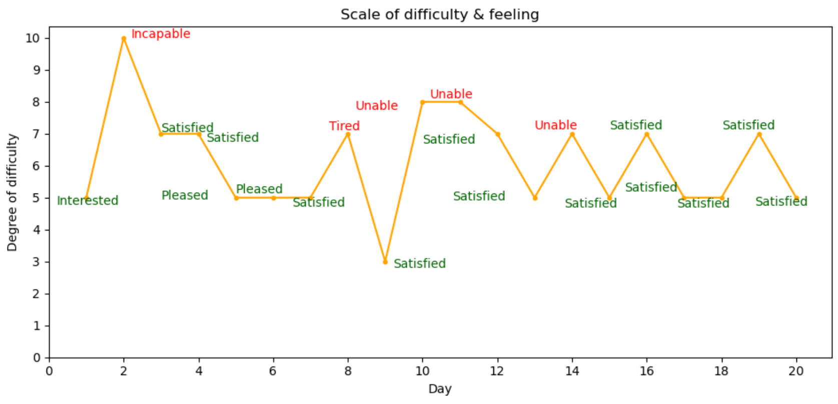
Typing speed. The average typing speed in Switch mode was 1.85 seconds (Fig. 5.3). The range was from 1.17 seconds to 3.1 seconds. Fig. 5.16 shows the typing speed of all characters in Switch mode. It is noticeable here, that the five slowest characters require many fingers to type. The reason for the high typing speed could be the problem with the battery. A low battery level means that the sensors furthest away from the thumb (little finger and ring finger) are not recognised properly. Overall, the typing speed is not very reliable because of the problems on day 19 and 20.

The words per minute here is 6.3 wpm.

Summary. The problems in switch mode were that the characters 'ArrowUp' and 'ArrowDown' were not distinguishable within the bubbles and a key combination was stored incorrectly in the database. On the last two days, the wrong characters were output despite the correct tap combination. This can be attributed to the low battery level of the Tap Strap. The battery level of the device does not match the battery level displayed on the computer. Despite the problems, the participant states that the switch mode is easy to learn.



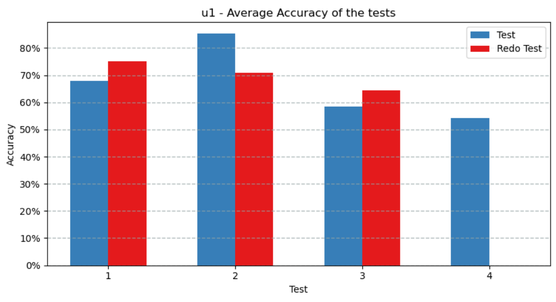
(a) u1 - *Difficulty Scale*.



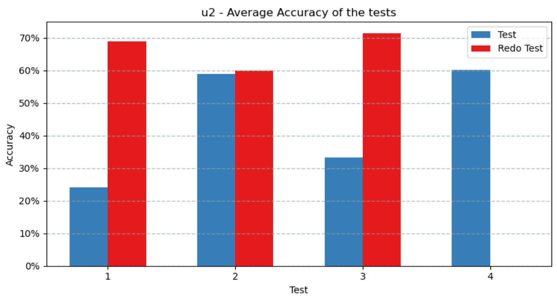
(b) u2 - *Difficulty Scale*.

Figure 5.17: The graphs above represent the *Difficulty Scale* of the daily learning session with the selected feeling of the study participant for the session. Positive feelings are green and negative feelings are red.

Difficulty Scale. The *Difficulty Scale* is part of the *Daily Questionnaire* and is based on the answer to the question "On a scale of 1-10, how difficult did you find your exercise session today? (Easy = 1; Difficult = 10)". Fig. 5.17a shows the progression of u1 with the annotated feeling of the session and Fig. 5.17b shows the scale of u2. The selection criteria for choosing the number and the feeling were often how well they could remember the combinations, how often they switched to a wrong mode and whether *Learn2Tap* ran smoothly. First, the progression of the scale of u1 is discussed in more detail. On day 1, the participant had difficulty connecting her Tap Strap to the computer because the device disconnected more often due to low battery. The dissatisfaction on day 3 is due to switching to switch mode. This problem occurred less frequently on day 4, hence the choice of 2. The increase on day 5 is due to the participant having problems remembering the combinations. The ups and downs between 5-10 depend on how well she remembered the combinations and whether she frequently switched to a wrong mode. On day 11, u1 was unable to complete any learning sessions because the 'Cap Lock' key on the laptop keyboard was activated, causing the Tap Strap to output incorrect combinations. On day 15, u1 was frustrated due to difficulties in performing the Triple-Tap and difficulties in remembering the key combinations. From day 17, u1 practised the Switch mode which she found easy. The small outlier upwards on day 19 was due to



(a) u1 - Accuracy.



(b) u2 - Accuracy.

Figure 5.18: The above diagrams show the accuracy of the individual weekly tests.

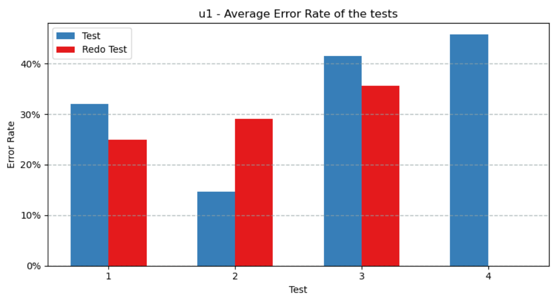
the Tap Strap not working correctly. The course of u2 has the first outlier upwards on day 2, here the participant feels incapable because she switches to Switch mode more often. After that, the scale settles between 5 and 7 until day 8. Depending on how well she could remember the new combinations. On day 9, the *Difficulty Scale* is a 3. On this day, the learning application crashed several times, so that no difficulties could occur when learning new combinations. The increasing value on day 8 is due to learning the combination ● ○ ● ● ● in Double-Tap mode. Since inaccurate typing leads to switching to switch mode with changing the mapping. The remaining days are between 5 and 7 on the *Difficulty Scale*, for the same reason as between day 3 and 8, how well she was able to remember the new combinations.

5.2.2 Weekly test

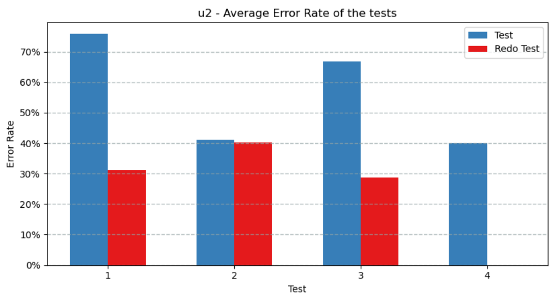
Once a week, the study participants were given a test (see Fig. 5.1b), which contained characters that had already been learned and were based on programming with Pharo. In the *Last Session*, the participants repeated tests 1, 2 and 3 in addition to 'Test 4'. In the following, the results of the tests are presented, classified into the categories accuracy, error rate, typing speed and words per minute. The detailed results of the accuracy and typing speed of the characters used in the test can be seen in the appendix (A.6).

Accuracy. The accuracy of the letters is calculated as follows:
$Accuracy = timesShowen/numberOfAttempts$. If a letter is skipped with the right arrow key, it is included in the final result with an accuracy of 0%. When we look at the diagrams from Fig. 5.18, we notice that u1 achieved the highest accuracy in 'Test 2' with 85% and the lowest in 'Test 4' with about 55%. U2 achieved the highest accuracy in 'Redo Test 3' with 72% and the lowest in 'Test 1' with 25%. In the case of u2, a significantly higher accuracy was achieved in the repeatability test. The low accuracy in 'Test 1' and 'Test 3' is because she switched to other modes in these tests, which resulted in many errors and a large part of the time of the test was spent trying to switch back to the correct mode. The tests are only a sample of the participants skills, but the overall accuracy is relatively low, with an average of 68.14% for u1 and 54.14% for u2.

Error rate. The error rate is calculated in the same way as for *Learn2Tap*:
$ErroRate = 100 - Accuracy$. Accordingly, the highest error rate for u1 is 45% for 'Test 4' and for u2 is 75% for 'Test 1'. The lowest error rate for u1 is 15% for 'Test 2' and for u2 is 28% for 'Redo Test 3' (Fig. 5.19). The average error rate is 31.86% for u1 and 45.86% for u2. Looking at the

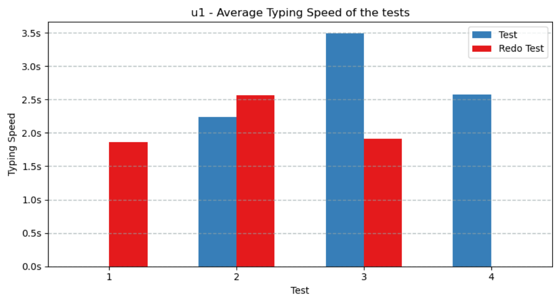


(a) u1 - Error rate.

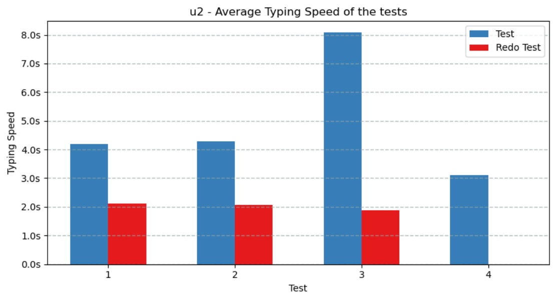


(b) u2 - Error rate.

Figure 5.19: The above diagrams show the error rate of the individual weekly tests.



(a) u1 - Typing speed.



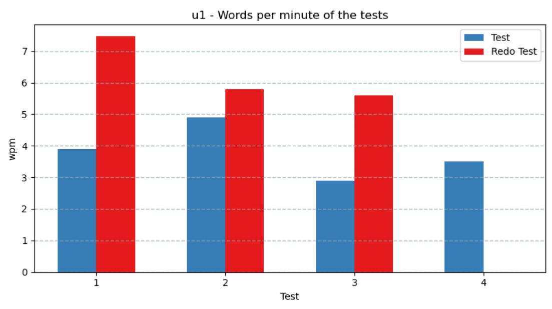
(b) u2 - Typing speed.

Figure 5.20: The above diagrams show the typing speed of the individual weekly tests.

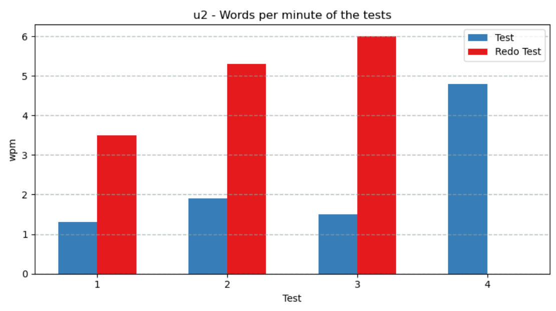
diagrams from the appendix A.6 on the accuracy of the individual characters, it is noticeable that characters like '%' and '&' have a lower average accuracy and thus a higher error rate. This may be caused by the fact that characters are created by Triple- and Double-Taps. Which are difficult to execute, according to the study participants. Moreover, these two modes were only learned later, so that the participants found it more difficult to remember further characters.

Typing speed. When calculating the typing speed, only the characters that were not skipped using 'rightArrow' were calculated. U1 achieved the best typing speed in the repetition of 'Test 1' with 1.9 seconds and u2 in the repetition of test number 2 with 2.05 seconds on average. The worst typing speed was achieved by u1 and u2 in 'Test 3' with 3.5 seconds and 8.1 seconds respectively (see Fig. 5.20). The reason for u2 is that she switched to shift or uppercase mode several times. If the typing speed is good, the accuracy is higher in most cases. For u1 the average typing speed of all tests is 2,44 sec and for u2 3,64 sec. When we look at the diagrams of the typing speed for individual characters from the appendix (see A.6), we notice that many tests have one or two outliers with a very high typing speed. For example, u1 'Redo Test 2', 'r' = 6.5 seconds or u2 'Test 4', '.' = 15 seconds. These outliers are characters with combinations similar to those used to switch to another mode. With 'r' (○ ● ● ● ●), inaccurate taps can switch to Switch mode. And with Double-Tap '.', the combination could be mixed up with the combination from capitalisation (○ ● ○ ○ ○).

Words per minute. The words per minute is an indication that can be used to compare the typing speed of different input devices. The typing speed of the fixed keyboard is between 20-40 wpm or 40-60 wpm, depending on the user's ability [HT04]. When Fig. 5.21 is viewed, the best value for u1 was achieved in 'Redo Test 1' with 7.5 wpm and for u2 in 'Test 3' with 6 wpm. The average



(a) u1 - Words per minute.



(b) u2 - Words per minute.

Figure 5.21: The above diagrams show the words per minute of the individual weekly tests.

wpm value for all tests is 4.87 wpm for u1 and 3.44 wpm for u2. However, compared to other input devices, the values are rather disappointing (see section 2). The values for words per minute are derived from the typing speed, so the argument for these values is the same as for typing speed. The test results that should be looked at first are those of 'Test 3' and 'Test 4', as the participants had to tap Pharo Code in these tests. If we look at the accuracy of 'Test 4', we notice that both participants did not know characters such as the bracket (0% accuracy) (see appendix A.6). These are an important part of Pharo. The result of 'Redo Test 3' is positively striking; both participants (5.5 wpm/ 6 wpm) achieved a good number of words per minute for their circumstances. This test was performed as the last of the four tests on the final day, which means that the participants had practised text tapping with the Tap Strap beforehand. However, u1 had problems with characters here too, u2 knew the combination of all characters except the 'l'. None of the participants typed the code snippet completely. This means that at the time of the test they were not able to type four short lines of Pharo code within 2 minutes. The conclusion of the test results is that with further practice the participants should be able to type Pharo code, but much slower than with a fixed keyboard. Furthermore, when programming with the Tap Strap, the participants first must think about which combination results in a certain character, so that the participants are distracted from creating the code and the Tap Strap represents an additional mental effort.

Conclusion of weekly tests. The tests serve as a snapshot of the user's skills. The values of the individual characters in relation to typing speed and accuracy are not particularly meaningful, as many characters were only used once in a test. Overall, the accuracy in the repeat tests was better in most cases. In addition, the value of the words per minutes has increased. From this it can be assumed that the participants have become better during the study. If one compares the values of the test (accuracy, typing speed) with those of a fixed keyboard, the Tap Strap performs poorly.

5.2.3 Discussion

When we look at the typing speed results in connection with accuracy, the Tap Strap performs poorly compared to other solutions for programming in AR or VR. In general, many conceptual issues occurred with the Tap Strap during the 20 days. The most common problem was switching to another mode without noticing it. This is supported by u1's statement on day 11, among others: "I accidentally switched modes and mappings and could not really practise anything new". The possibility to give the user feedback on which mode they are in is difficult, as the Tap Strap does not give any information about this, and no key combination is given when switching to another mode.

One of the points u1 mentioned as an improvement in the second weekly questionnaire was “That it would be helpful if the app tells you if you switched the mode”. Furthermore, the participants found learning all the combinations and distinguishing between them tedious. This is also one of the main problems in learning the Tap Strap combinations, because as the learning of the combinations progresses, the participants forget the combinations that were learned earlier. After all, there are over 100 combinations to learn. In the final interview, they both stated that they would prefer a fixed keyboard to a Tap Strap even in VR, as it is hard to remember all the combinations. Both participants found the Double- and Triple-Tap mode difficult because the speed of the multiple taps must be correct. U2 mentioned on day 18 that it is difficult to hit the exact speed in Triple-Tap mode and that she often does more than 3 taps and ends up with a wrong character. Further, there are problems with the hardware of the Tap Strap. One of the problems is that the battery level of the Tap Strap on the computer does not correspond to the correct battery level. This leads to the Tap Strap no longer working properly and the user is not directly aware of the reason. Another problem was that one of the sensors on one of the Tap Straps broke and the device stopped working. The Tap Strap was in use for less than one hour on average over the course of 4 months. This could be an isolated case, but it did occur. One more issue that came up during the preparation of the study, which is mentioned in section 3.1.1, is that it is not possible to change the mapping of the Tap Strap with the Android app ’TapManager’. As the app tells the user that the Tap Strap needs an update to download a personalised mapping, at the same time the app tells the user that there are currently no updates for the Tap Strap. In the end, neither of the participants felt capable of programming with the Tap Strap in VR.

The feedback on *Learn2Tap* was predominantly positive. The users found the application to have good usability and they would use it again if they had to learn the combinations again. The participants appreciated the structure of the *Learning Units*, as it is simple and the levels with the falling bubbles teach the combinations in a playful way. The games also helped the participants not to forget the combinations. One point of criticism was the display of certain characters in the bubbles (’ArrowUp’, ’ArrowDown’,’_’, ’-’), and that in the *Time To Tap* game one attempt to enter a character is not enough. As for the structure of the mapping, u1 suggested that it could be structured alphabetically and that characters that have two combinations (e.g.. ’w’ and ’j’) only keep the simpler of the two combinations, so that the learning effort is reduced. To summarise the results of the study, the participants would not use the Tap Strap as a keyboard. However, if they had to use it, the participants would use *Learn2Tap* again to learn the combinations.

6 Conclusion and Outlook

This concluding chapter is intended to review and critically examine the project and the work that has been done. First, the results of the thesis is summarised. Finally, an outlook is given, showing what a possible expansion could look like and what opportunities could arise from it.

6.1 Conclusion and outcomes

Most of the problems that occurred during the preparation and evaluation of the user study, lead to the conclusion that the Tap Strap is unsuitable as a text input device. The first problem occurred when using the Android app 'TapManager'. Here, the self-created mapping for *Learn2Tap* could not be downloaded. No problems occurred when using the iOS app. The most common problem during the study was that users accidentally switched to the wrong tap mode (Switch mode, Shift mode, uppercase mode) or changed the mapping. Other problems related to the hardware of the Tap Strap were that if the keyboard layout on the laptop was not set to US keyboard layout, incorrect characters were output. Also, activating the 'Cap Lock' key on the laptop keyboard results in the output of incorrect characters. The actual battery level of the Tap Strap and the displayed battery level do not match. A higher battery level is displayed than the actual battery level. If the Tap Strap is insufficiently charged, the sensors furthest away from the thumb no longer react and incorrect combinations are output, despite correct tapping. When evaluating the results of the studies, it is noticeable that the overall accuracy is low. One of the main problems is remembering the different combinations. There are over 100 combinations in total. This also leads to a relatively poor typing speed, as users first have to remember what the correct combination is and then position their fingers to tap the combination. In Double- and Triple-Tap mode, the users have to hit the exact tap speed, otherwise the Tap Strap does not recognise them as double or triple taps.

All in all, *Learn2Tap* has proven itself as a learning application. The study participants found using the web application to learn the combinations enjoyable and beneficial. The biggest criticism of *Learn2Tap* was that some characters, such as '-' and '_' or 'ArrowUp' and 'ArrowDown', were difficult to distinguish within the bubbles.

Overall, it can be concluded that the Tap Strap is not convincing as a text input device for on-the-go visualisation. This is also supported by the statement of the study participants that both would prefer to use a real keyboard in VR, even though they cannot see the keys, instead of using the Tap Strap. The main reason that the Tap Strap performs so poorly is that learning the combinations is tedious and the problems listed above make it even more difficult to use. However, if the combinations of the Tap Strap had to be learned, *Learn2Tap* would be a suitable learning application.

Despite the criticism of the Tap Strap, there are possibilities to improve the user experience. These are explained in the following section.

6.2 Outlook and opportunities

As mentioned in the section above, one of the main problems of the Tap Strap is the output of wrong characters, where the cause is not directly obvious to the user. Therefore, a useful extension of the Tap Strap would be to provide feedback to the user. For example, that ’Cap Lock’ is activated, that the keyboard layout does not match the selected layout of the mapping and that the correct battery level is displayed. It would also be helpful if the user could determine which mode the Tap Strap is currently in (Normal, Shift or Switch), for example, by changing the colour of the light on the thumb ring depending on the mode. A further extension would be that the user can set which mode they are currently in by pressing a button on the keyboard. This would prevent the user from accidentally changing the mode by inaccurate tapping. The problem with the configuration via the Android app ’TapManger’ can easily be solved if it adapts the functions of the iOS app. Furthermore, the Tap Strap could include an intelligent system that learns the ’tap style’ of the user and adapts the output to this. For example, the input speed of the Triple-Tap could be adjusted by the user.

Improvement points of *Learn2Tap* are among others the integration of an algorithm that unlocks further characters in the *Learning Units* depending on the user’s ability, similar to Keybr (see section 2.1). Another function of the algorithm should be that it evaluates the accuracy of the already learned characters and the characters with a lower accuracy are repeated more often. Another point of improvement is that when creating a mapping for AVAR, characters that have two combinations (e.g. ’w’ and ’c’) only keep the simpler combination. This reduces the learning load.

As a thought point and possible research proposal, it would be worth analysing whether the Tap Strap is a suitable input tool for people who cannot use an fixed keyboard or virtual keyboard due to physical limitations.

Bibliography

[03a] *FrogPad*. Accessed: 2021-08-03. URL: http://www.frogpad.com/FPInfo-Training.html (cit. on pp. 15, 17, 18).

[03b] *KeyBr*. Accessed: 2021-08-03. URL: http://twiddler.tekgear.com/tutor/twiddler.html?__cf_chl_managed_tk__=pmd_ecb3930e2c2f24ffd8b8996ac1e14515773ab8aa-1628002448-0-gqNtZGzNAvijcnBszQPi (cit. on p. 18).

[03c] *Twiddler Website*. Accessed: 2021-08-03. URL: https://twiddler.tekgear.com/ (cit. on p. 15).

[14a] *CyKey Keyboard*. Accessed: 2021-09-14. URL: https://www.latentexistence.me.uk/wp-content/uploads/2012/03/CyKey.jpg (cit. on pp. 17, 19).

[14b] *Decatxt Keyboard*. Accessed: 2021-09-14. URL: https://gulfcoastmakers.files.wordpress.com/2018/03/hero31.png (cit. on p. 17).

[26a] *CyKey*. Accessed: 2021-08-26. URL: https://sites.google.com/site/cykeybellaire/ (cit. on pp. 18, 19).

[26b] *TapStrap*. Accessed: 2021-08-26. URL: https://www.tapwithus.com/ (cit. on pp. 18, 53).

[Azu97] R. T. Azuma. "others,"A survey of augmented reality,"" in: *Presence-Teleoperators and Virtual Environments* 6.4 (1997), pp. 355–385 (cit. on p. 15).

[BK19] C. Boletsis, S. Kongsvik. "Controller-based text-input techniques for virtual reality: An empirical comparison". In: *International Journal of Virtual Reality (IJVR)* 19.3 (2019) (cit. on pp. 20, 21).

[CG11] K. Chalil Madathil, J. S. Greenstein. "Synchronous remote usability testing: a new approach facilitated by virtual worlds". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2011, pp. 2225–2234 (cit. on p. 58).

[DBWK19] J. Dudley, H. Benko, D. Wigdor, P. O. Kristensson. "Performance envelopes of virtual keyboard text input strategies in virtual reality". In: *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2019, pp. 289–300 (cit. on p. 21).

[FZ94] G. Forman, J. Zahorjan. "The challenges of mobile computing". In: *Computer* 27.4 (1994), pp. 38–47. DOI: 10.1109/2.274999 (cit. on pp. 3, 15).

[HB13] U. Hammerschall, G. H. Beneken. *Software requirements*. Deutsch. Literaturverzeichnis Seite 419-426. München, 2013. URL: https://ebookcentral.proquest.com/lib/uni-stuttgart/detail.action?docID=5583740 (cit. on p. 29).

[HT04] C. Houser, P. Thornton. "Japanese college students' typing speed on mobile devices". In: *The 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education, 2004. Proceedings*. IEEE. 2004, pp. 129–133 (cit. on pp. 20, 75).

[KPG17] S. Karthika, P. Praveena, M. GokilaMani. “Hololens”. In: *International Journal of Computer Science and Mobile Computing* 6.2 (2017), pp. 41–50 (cit. on p. 20).

[KSF+18] P. Knierim, V. Schwind, A. M. Feit, F. Nieuwenhuizen, N. Henze. “Physical keyboards in virtual reality: Analysis of typing performance and effects of avatar hands”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–9 (cit. on pp. 20, 21).

[Lav15] S. M. Lavalle. *Virtual reality*. 2015 (cit. on p. 15).

[LLY+19] L. H. Lee, K. Y. Lam, Y. P. Yau, T. Braud, P. Hui. “Hibey: Hide the keyboard in augmented reality”. In: *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*. IEEE. 2019, pp. 1–10 (cit. on pp. 15, 20).

[LSP+04] K. Lyons, T. Starner, D. Plaisted, J. Fusia, A. Lyons, A. Drew, E. Looney. “Twiddler typing: One-handed chording text entry for mobile phones”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2004, pp. 671–678 (cit. on p. 18).

[MSY+20] L. Merino, B. Sotomayor-Gómez, X. Yu, R. Salgado, A. Bergel, M. Sedlmair, D. Weiskopf. “Toward agile situated visualization: An exploratory user study”. In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–7 (cit. on pp. 15, 27).

[RBS78] N. Rochester, F. Bequaert, E. Sharp. “The Chord Keyboard”. In: *Computer* 11.12 (1978), pp. 57–63. DOI: 10.1109/C-M.1978.218024 (cit. on p. 18).

[Shn98] B. Shneiderman. *Designing the User Interface – Strategies for Effective Human-Computer-Interaction*. 3rd ed. Addison-Wesley Longman, 1998. ISBN: 0201694972 (cit. on pp. 29, 30).

[ZHS02] S. Zhai, M. Hunter, B. A. Smith. “Performance optimization of virtual keyboards”. In: *Human–Computer Interaction* 17.2-3 (2002), pp. 229–269 (cit. on p. 66).

All links were last followed on September 09, 2021.

A Appendix

The appendix contains statistics on the frequency of symbols appearing in Pharo, an overview of the mapping created for AVAR and code extracts from *Learn2Tap*. Furthermore, the forms of the user study and the questionnaires are included. There are also detailed diagrams of the data collected in the user study on the weekly tests.

A.1 Frequency of the Pharo Characters

In this section you will find the diagrams that give an overview of how often certain characters appear in Pharo.

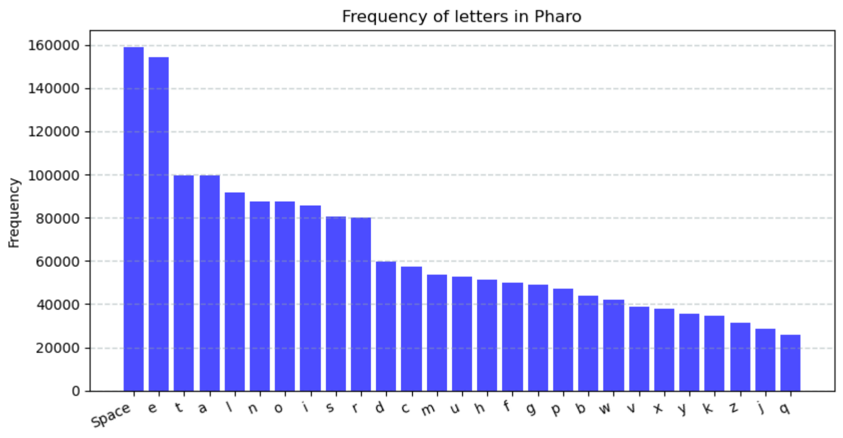


Figure A.1: The diagram shows the frequency with which individual letters appear in Pharo.

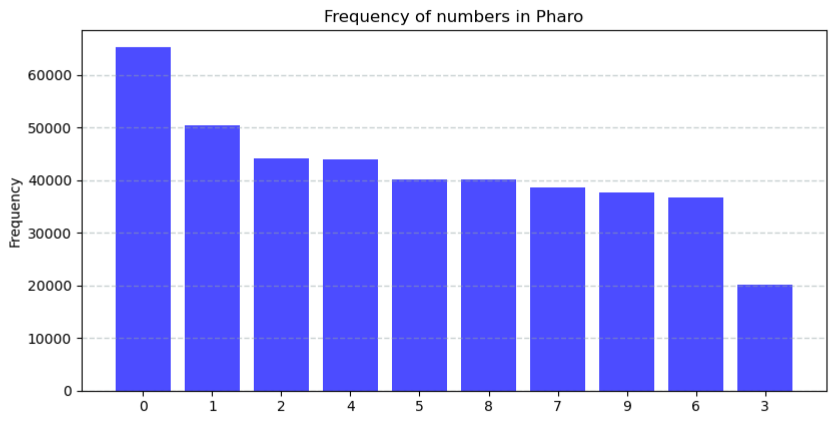


Figure A.2: The diagram above shows the frequency with which numbers appear in Pharo.

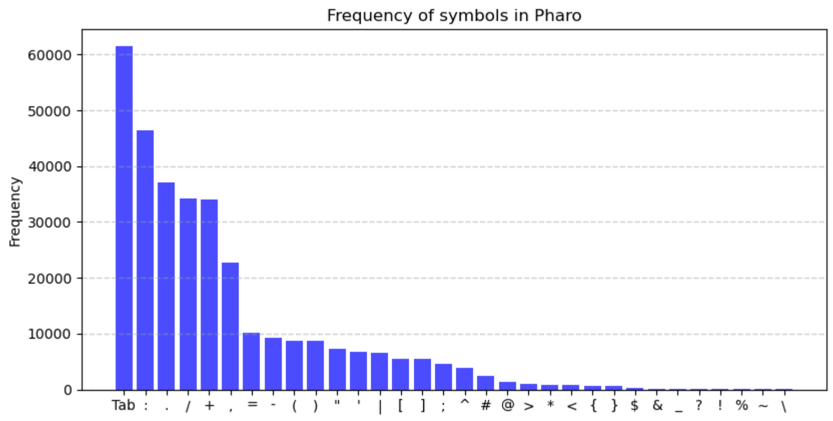


Figure A.3: The diagram shows the frequency with which symbols appear in Pharo.

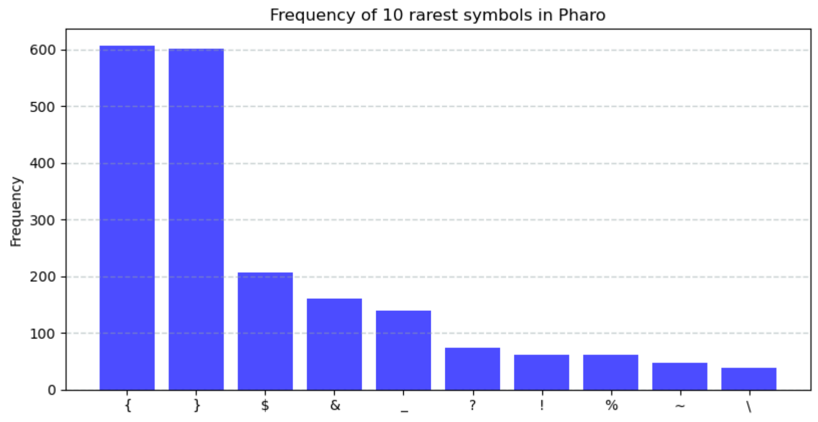
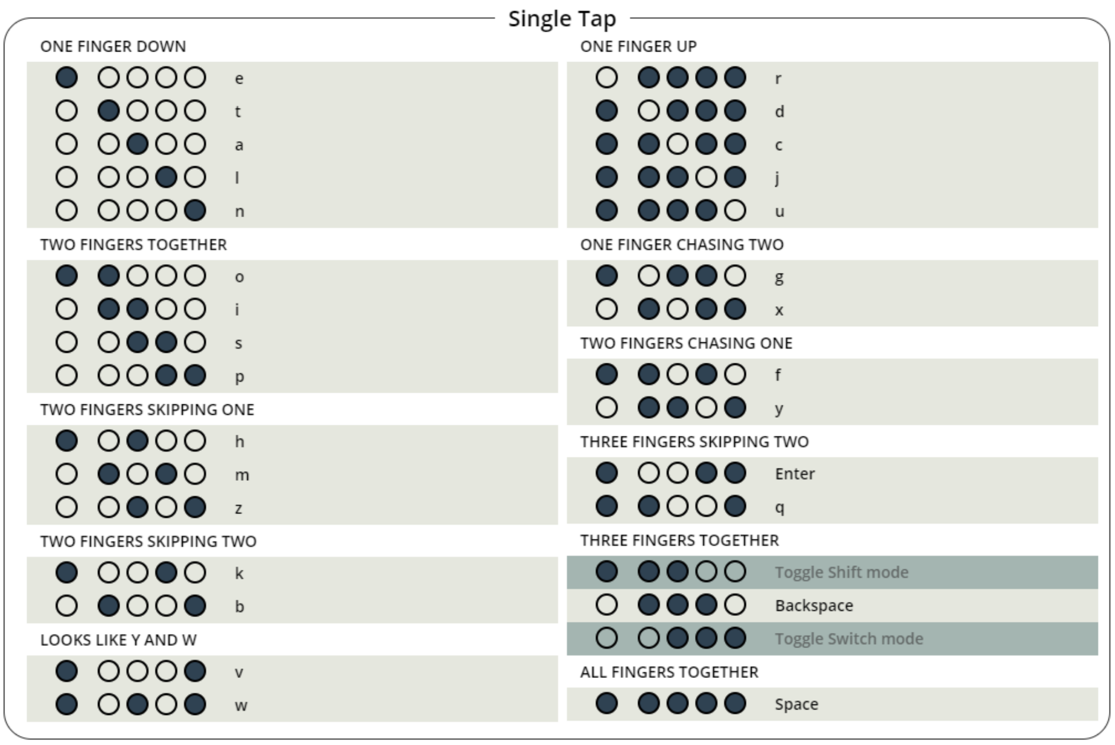


Figure A.4: The chart above shows the frequency of the 10 symbols that appear most rarely in Pharo.

A.2 Tap Strap Mapping

Here I show in detail the Tap Strap mapping designed for AVAR.



Single Tap

ONE FINGER DOWN

● ○○○○	e
○ ●○○○	t
○ ○●○○	a
○ ○○●○	l
○ ○○○●	n

TWO FINGERS TOGETHER

● ●○○○	o
○ ●●○○	i
○ ○●●○	s
○ ○○●●	p

TWO FINGERS SKIPPING ONE

● ○●○○	h
○ ●○●○	m
○ ○●○●	z

TWO FINGERS SKIPPING TWO

● ○○●○	k
○ ●○○●	b

LOOKS LIKE Y AND W

● ○○○●	v
● ○●○●	w

ONE FINGER UP

○ ●●●●	r
● ○●●●	d
● ●○●●	c
● ●●○●	j
● ●●●○	u

ONE FINGER CHASING TWO

● ○●●○	g
○ ●○●●	x

TWO FINGERS CHASING ONE

● ●○●○	f
○ ●●○●	y

THREE FINGERS SKIPPING TWO

● ○○●●	Enter
● ●○○●	q

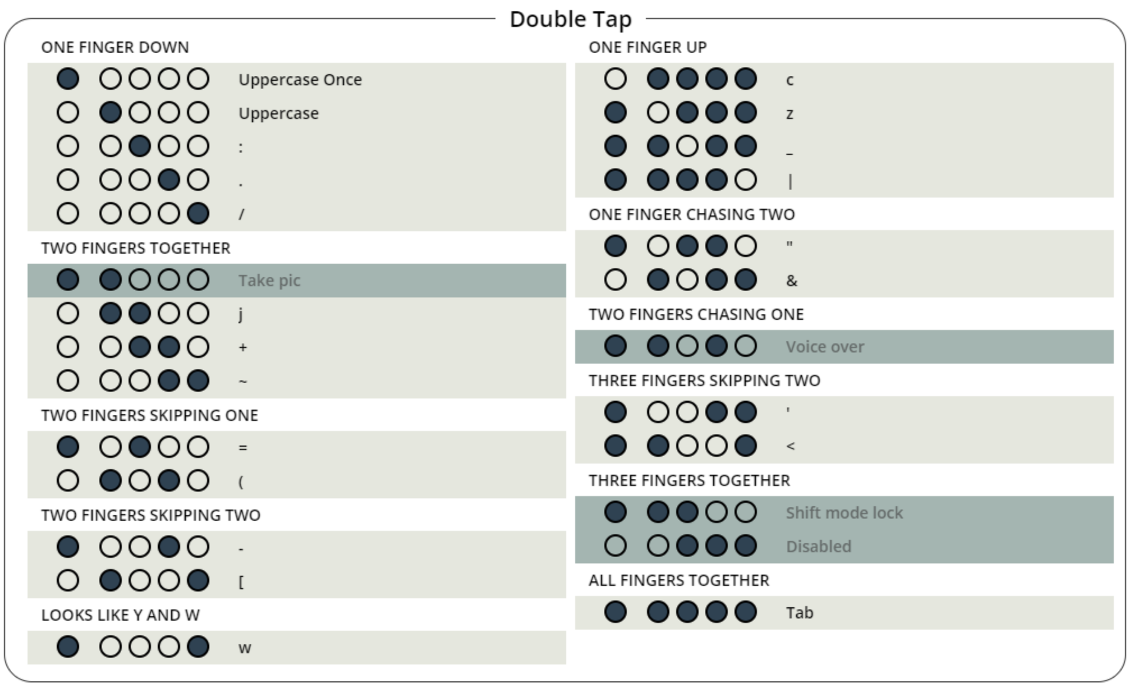
THREE FINGERS TOGETHER

● ●●○○	Toggle Shift mode
○ ●●●○	Backspace
○ ○●●●	Toggle Switch mode

ALL FINGERS TOGETHER

● ●●●●	Space

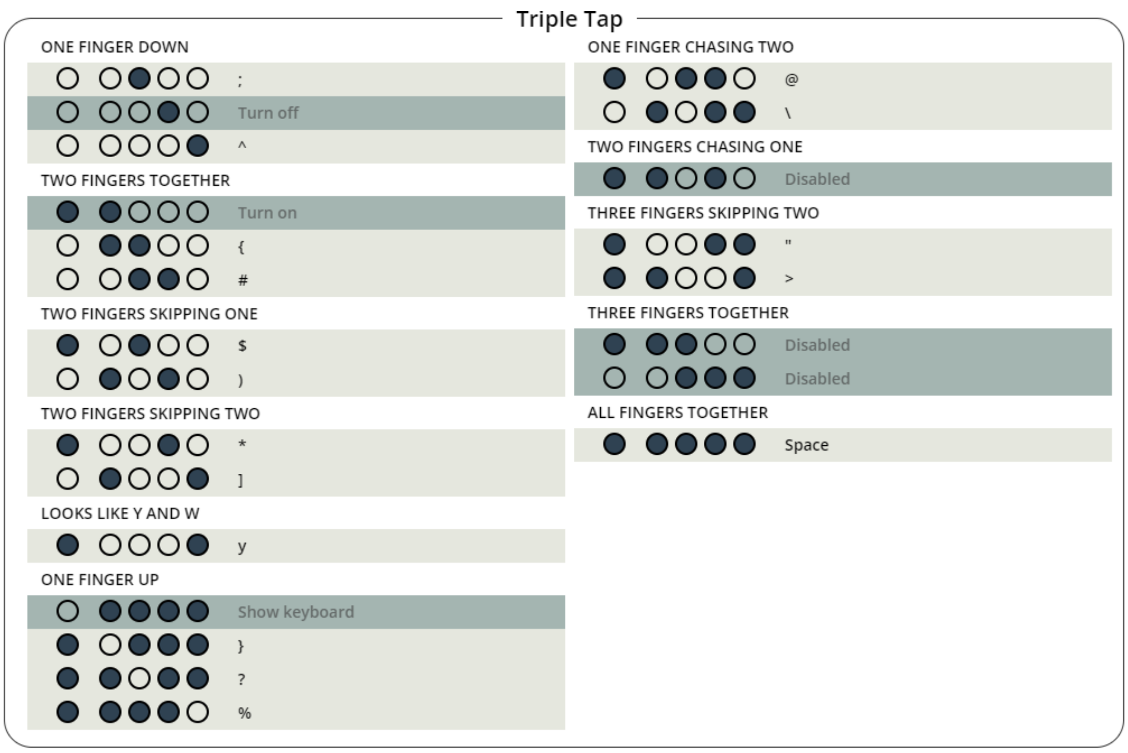
Figure A.5: This Figure shows the Tap Strap mapping for Single-Tap mode.



Double Tap

ONE FINGER DOWN	
● ○○○○	Uppercase Once
○ ●○○○	Uppercase
○ ○●○○	:
○ ○○●○	.
○ ○○○●	/
TWO FINGERS TOGETHER	
● ●○○○	Take pic
○ ●●○○	j
○ ○●●○	+
○ ○○●●	~
TWO FINGERS SKIPPING ONE	
● ○●○○	=
○ ●○●○	(
TWO FINGERS SKIPPING TWO	
● ○○●○	-
○ ●○○●	[
LOOKS LIKE Y AND W	
● ○○○●	w
ONE FINGER UP	
○ ●●●●	c
● ○●●●	z
● ●○●●	_
● ●●●○	|
ONE FINGER CHASING TWO	
● ○●●○	"
○ ●○●●	&
TWO FINGERS CHASING ONE	
● ●○●○	Voice over
THREE FINGERS SKIPPING TWO	
● ○○●●	'
● ●○○●	<
THREE FINGERS TOGETHER	
● ●●○○	Shift mode lock
○ ○●●●	Disabled
ALL FINGERS TOGETHER	
● ●●●●	Tab

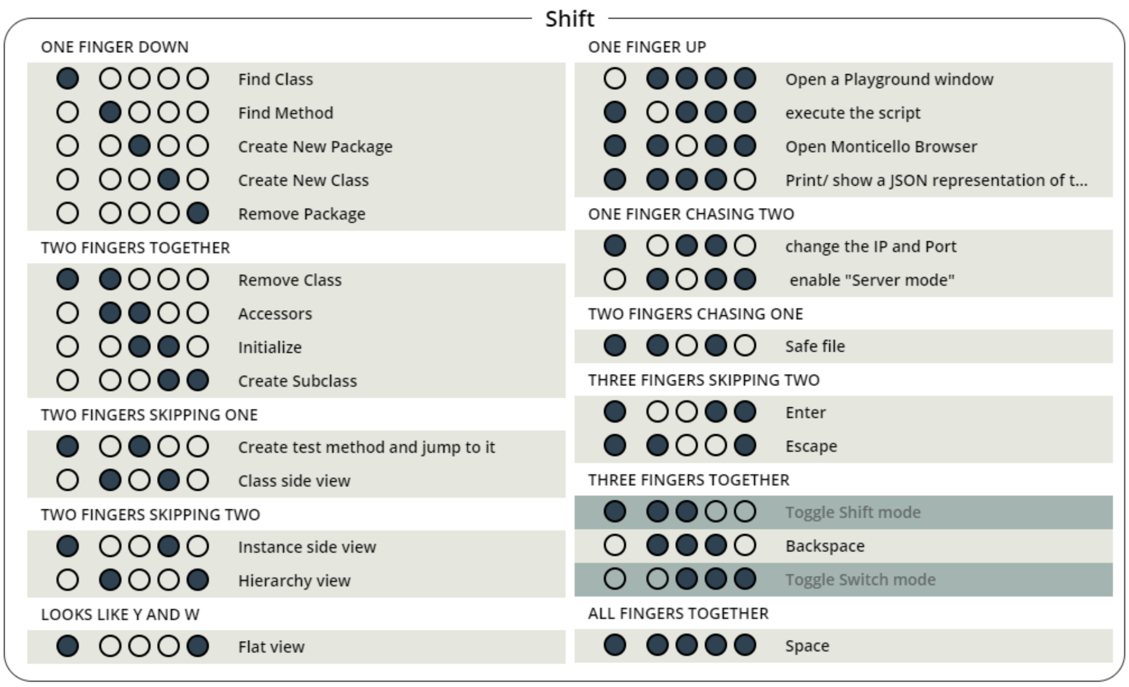
Figure A.6: This Figure illustrates the Tap Strap Mapping for Double-Tap mode.



Triple Tap

ONE FINGER DOWN	
○ ○●○○	;
○ ○○●○	Turn off
○ ○○○●	^
TWO FINGERS TOGETHER	
● ●○○○	Turn on
○ ●●○○	{
○ ○●●○	#
TWO FINGERS SKIPPING ONE	
● ○●○○	$
○ ●○●○	)
TWO FINGERS SKIPPING TWO	
● ○○●○	*
○ ●○○●	]
LOOKS LIKE Y AND W	
● ○○○●	y
ONE FINGER UP	
○ ●●●●	Show keyboard
● ○●●●	}
● ●○●●	?
● ●●●○	%
ONE FINGER CHASING TWO	
● ○●●○	@
○ ●○●●	\
TWO FINGERS CHASING ONE	
● ●○●○	Disabled
THREE FINGERS SKIPPING TWO	
● ○○●●	"
● ●○○●	>
THREE FINGERS TOGETHER	
● ●●○○	Disabled
○ ○●●●	Disabled
ALL FINGERS TOGETHER	
● ●●●●	Space

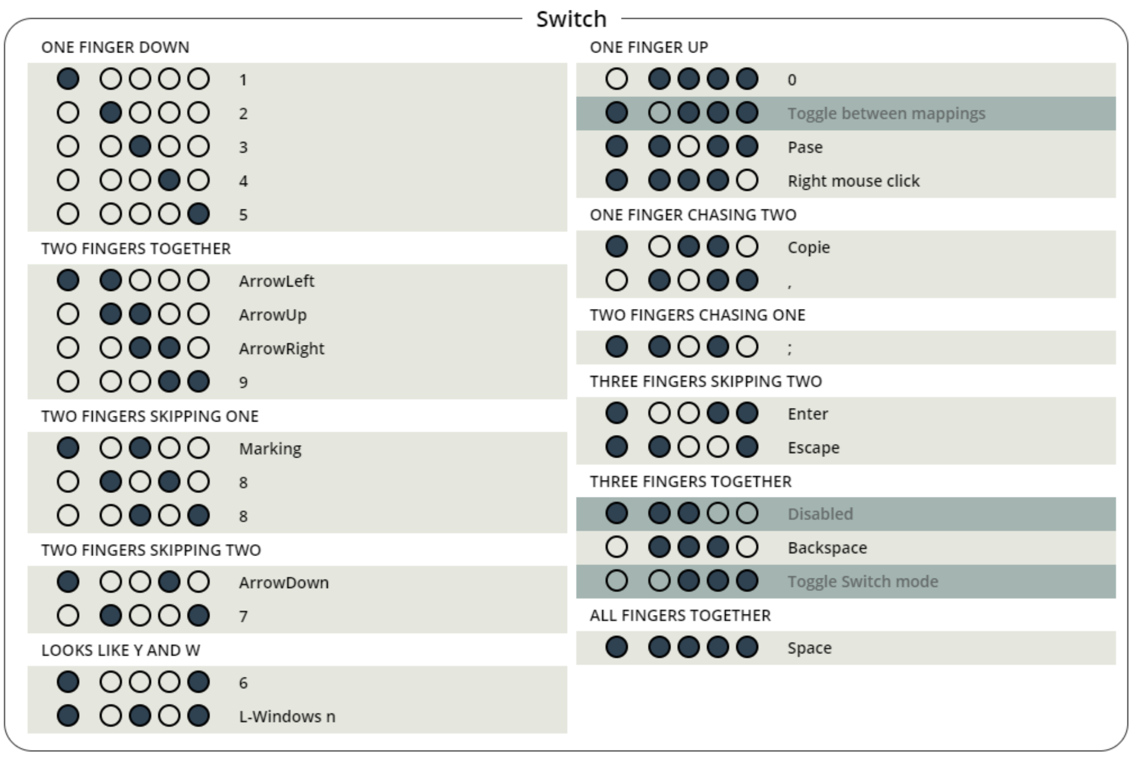
Figure A.7: The Figure above provides the Tap Strap Mapping for the Triple-Tap mode.



Shift

ONE FINGER DOWN	
● ○○○○	Find Class
○ ●○○○	Find Method
○ ○●○○	Create New Package
○ ○○●○	Create New Class
○ ○○○●	Remove Package
TWO FINGERS TOGETHER	
● ●○○○	Remove Class
○ ●●○○	Accessors
○ ○●●○	Initialize
○ ○○●●	Create Subclass
TWO FINGERS SKIPPING ONE	
● ○●○○	Create test method and jump to it
○ ●○●○	Class side view
TWO FINGERS SKIPPING TWO	
● ○○●○	Instance side view
○ ●○○●	Hierarchy view
LOOKS LIKE Y AND W	
● ○○○●	Flat view
ONE FINGER UP	
○ ●●●●	Open a Playground window
● ○●●●	execute the script
● ●○●●	Open Monticello Browser
● ●●●○	Print/ show a JSON representation of t...
ONE FINGER CHASING TWO	
● ○●●○	change the IP and Port
○ ●○●●	enable "Server mode"
TWO FINGERS CHASING ONE	
● ●○●○	Safe file
THREE FINGERS SKIPPING TWO	
● ○○●●	Enter
● ●○○●	Escape
THREE FINGERS TOGETHER	
● ●●○○	Toggle Shift mode
○ ●●●○	Backspace
○ ○●●●	Toggle Switch mode
ALL FINGERS TOGETHER	
● ●●●●	Space

Figure A.8: The Tap Strap mapping of the Shift mode is displayed here.



Switch

ONE FINGER DOWN	
● ○○○○	1
○ ●○○○	2
○ ○●○○	3
○ ○○●○	4
○ ○○○●	5
TWO FINGERS TOGETHER	
● ●○○○	ArrowLeft
○ ●●○○	ArrowUp
○ ○●●○	ArrowRight
○ ○○●●	9
TWO FINGERS SKIPPING ONE	
● ○●○○	Marking
○ ●○●○	8
○ ○●○●	8
TWO FINGERS SKIPPING TWO	
● ○○●○	ArrowDown
○ ●○○●	7
LOOKS LIKE Y AND W	
● ○○○●	6
● ○●○●	L-Windows n
ONE FINGER UP	
○ ●●●●	0
● ○●●●	Toggle between mappings
● ●○●●	Pase
● ●●●○	Right mouse click
ONE FINGER CHASING TWO	
● ○●●○	Copie
○ ●○●●	,
TWO FINGERS CHASING ONE	
● ●○●○	;
THREE FINGERS SKIPPING TWO	
● ○○●●	Enter
● ●○○●	Escape
THREE FINGERS TOGETHER	
● ●●○○	Disabled
○ ●●●○	Backspace
○ ○●●●	Toggle Switch mode
ALL FINGERS TOGETHER	
● ●●●●	Space

Figure A.9: The Figure above provides the Tap Strap Mapping for the Switch mode.

A.3 Code Snippet

This section contains code extracts from the *Learn2Tap* web application.

Listing A.1 This code shows the appearance of a button in a menu of a mode.

```
//----------------DoubleTapMenu.js----------------------------------------
  <button class="normalButton" id="practicButtonMenuDT" type="button"
            onClick={() => {
              sendUserTracking(value, 'Button click', 'Unit 4 Button', 'Double Tap Menu' );
              getLevelValues([15,17,18,19,21], rightHand, 'Unit 4',levelName[3],4, false );
              Cookies.set('symbolIDsArray', [15,17,18,19,21])
              }
            }>
             <div class="row">
        <div class="col">
          <b>{levelName[3]}</b>
          <br></br>
          Unit 4
        </div>
        <div class="row">
          {starsTotalArr[3]}/9
          <img src={star} width="60px" height="100px" alt="star" />
             </div>
            </div>
  </button>
```

Listing A.2 This listing shows the 'nextShiftKeyCombi(combis, currLetterCount)' function.

```
//-------------------------------level.js----------------------------------------
 function nextShiftKeyCombi(combis, currLetterCount){

    var shiftKeyCombiCurr =combis[currLetterCount] + ',';
        var singleShiftKeys = [];
        var shiftKeysPressed = [];

        var currKeyI = "";
        for(var i = 0; i < shiftKeyCombiCurr.length; i++){
          var currSym = shiftKeyCombiCurr[i];


            if(currSym !== ' ' && currSym !==','){
              currKeyI = currKeyI + currSym;

            }else{
                singleShiftKeys.push(currKeyI);
              currKeyI = '';
              shiftKeysPressed.push(false);
            }
     }
    setCurrShiftKeys(singleShiftKeys);
    setReadShiftKey(shiftKeysPressed);
   }
```

Listing A.3 This code snippet established the connection between the server and the database.

```
const dbUser = mysql.createConnection({
    host:'localhost',
    user:'root',
    password: 'password',
    database:'tapstrapapp',

});
```

Listing A.4 This code snippet connects the server to a selected port, here port 3001.

```
//-------------------Server connection to the port--------------------------------
app.listen(3001, () =>{
    console.log("Your server is running on port 3001");
});
```

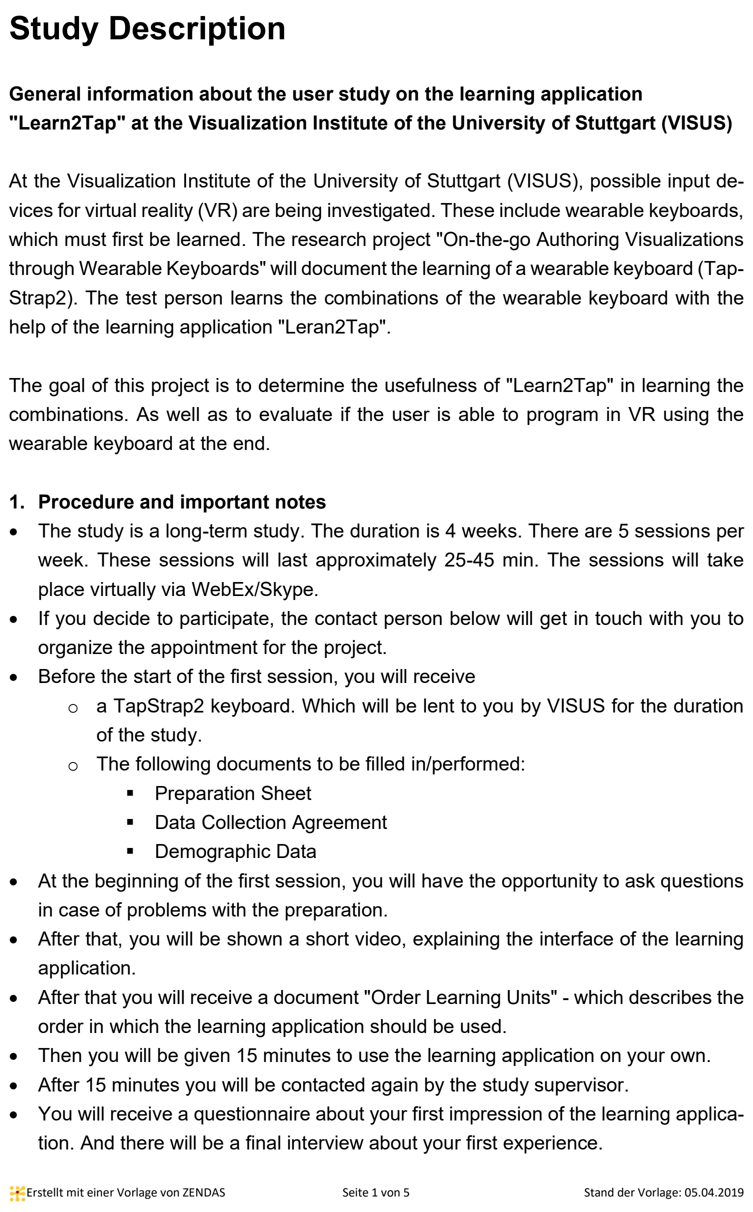
Listing A.5 Calculation and writing to the database of the new accuracy values of the Double-Tap mode.

```
app.post('/statisticDoubleTap',(req,res) =>{
      //read the data from the body

  //Read Accuracy data from database
     dbUser.query('SELECT * FROM doubletap WHERE userID=?',  id , (err, result) =>{
            if(err){
                console.log(err);
            }else{
        //Reading the database response
                const resultJson = Object.values(JSON.parse(JSON.stringify(result)))[0];
                const keys = Object.keys(resultJson);
                var valSids = [];
                keys.forEach((key, index) => {
                     if(key !== 'userID'){
                        valSids.push(resultJson[key]);
                       }
        });
        //Updating the Accuracy values
                for(var i = 0; i< letter.length ; i++){
          //Query if symbol was shown in level/game
                   if(resultStat[i] !== -1){
          //Query whether symbol in the database already has an accuracy value
          if(valSids[i] === 0){
            valSids[i]=resultStat[i];
                       }else{
                        valSids[i] =( valSids[i] + (2 * resultStat[i]))/3;
                      }
          }
         }

        //Saving the new accuracy values in the database
                dbUser.query('UPDATE doubletap SET sid_3=?, sid_4=?, sid_5=?,sid_7=?, sid_8=?,
 sid_9=?, sid_10=?, sid_11=?, sid_13=?, sid_14=?, sid_15=?, sid_17=?, sid_18=?, sid_19=?,
sid_21=?, sid_22=?, sid_23=?, sid_26=?, sid_27=?, sid_31=? WHERE userID = ?',
                    [valSids[0],valSids[1], valSids[2],valSids[3], valSids[4], valSids[5],
valSids[6], valSids[7], valSids[8],valSids[9], valSids[10],valSids[11],valSids[12],valSids
[13], valSids[14],valSids[15], valSids[16],valSids[17] ,valSids[18],valSids[19],id], (err,
result) =>{
                     if(err){
                         console.log(err);
                     }else{
                        console.log('success Double Tap Add Statistic to Database');
                     }
                })
             }
          })
    });
```

A.4 Forms of the User Study

This section contains the forms that the study participants received before the start of the study.

A.4.1 Consent Form



Study Description

General information about the user study on the learning application "Learn2Tap" at the Visualization Institute of the University of Stuttgart (VISUS)

At the Visualization Institute of the University of Stuttgart (VISUS), possible input devices for virtual reality (VR) are being investigated. These include wearable keyboards, which must first be learned. The research project "On-the-go Authoring Visualizations through Wearable Keyboards" will document the learning of a wearable keyboard (TapStrap2). The test person learns the combinations of the wearable keyboard with the help of the learning application "Leran2Tap".

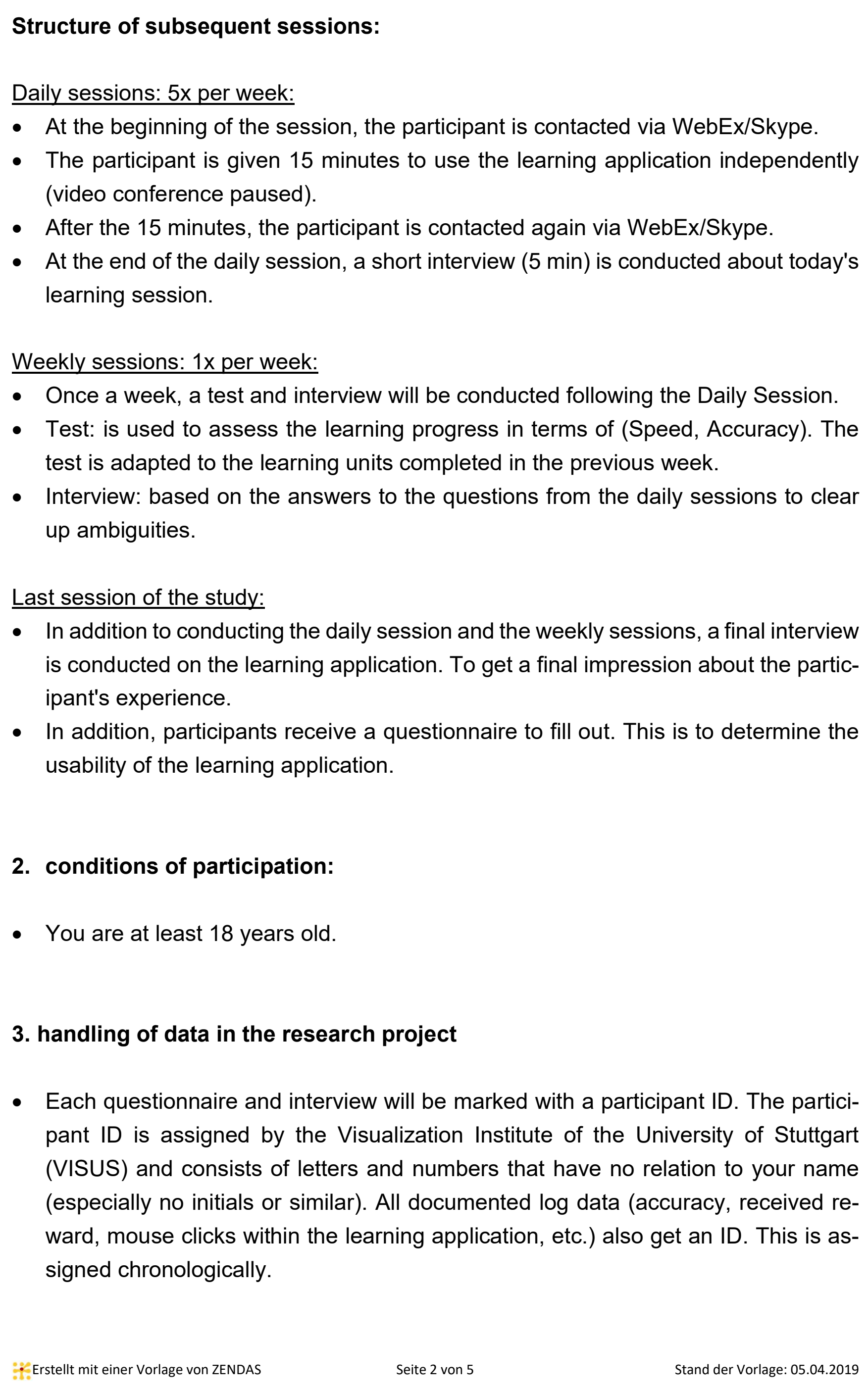
The goal of this project is to determine the usefulness of "Learn2Tap" in learning the combinations. As well as to evaluate if the user is able to program in VR using the wearable keyboard at the end.

1. Procedure and important notes

- The study is a long-term study. The duration is 4 weeks. There are 5 sessions per week. These sessions will last approximately 25-45 min. The sessions will take place virtually via WebEx/Skype.
- If you decide to participate, the contact person below will get in touch with you to organize the appointment for the project.
- Before the start of the first session, you will receive
    - a TapStrap2 keyboard. Which will be lent to you by VISUS for the duration of the study.
    - The following documents to be filled in/performed:
        - Preparation Sheet
        - Data Collection Agreement
        - Demographic Data
- At the beginning of the first session, you will have the opportunity to ask questions in case of problems with the preparation.
- After that, you will be shown a short video, explaining the interface of the learning application.
- After that you will receive a document "Order Learning Units" - which describes the order in which the learning application should be used.
- Then you will be given 15 minutes to use the learning application on your own.
- After 15 minutes you will be contacted again by the study supervisor.
- You will receive a questionnaire about your first impression of the learning application. And there will be a final interview about your first experience.

Erstellt mit einer Vorlage von ZENDAS Seite 1 von 5 Stand der Vorlage: 05.04.2019

Figure A.10: Page 1 - Consent Form.



Structure of subsequent sessions:

Daily sessions: 5x per week:

- At the beginning of the session, the participant is contacted via WebEx/Skype.
- The participant is given 15 minutes to use the learning application independently (video conference paused).
- After the 15 minutes, the participant is contacted again via WebEx/Skype.
- At the end of the daily session, a short interview (5 min) is conducted about today's learning session.

Weekly sessions: 1x per week:

- Once a week, a test and interview will be conducted following the Daily Session.
- Test: is used to assess the learning progress in terms of (Speed, Accuracy). The test is adapted to the learning units completed in the previous week.
- Interview: based on the answers to the questions from the daily sessions to clear up ambiguities.

Last session of the study:

- In addition to conducting the daily session and the weekly sessions, a final interview is conducted on the learning application. To get a final impression about the participant's experience.
- In addition, participants receive a questionnaire to fill out. This is to determine the usability of the learning application.

2. conditions of participation:

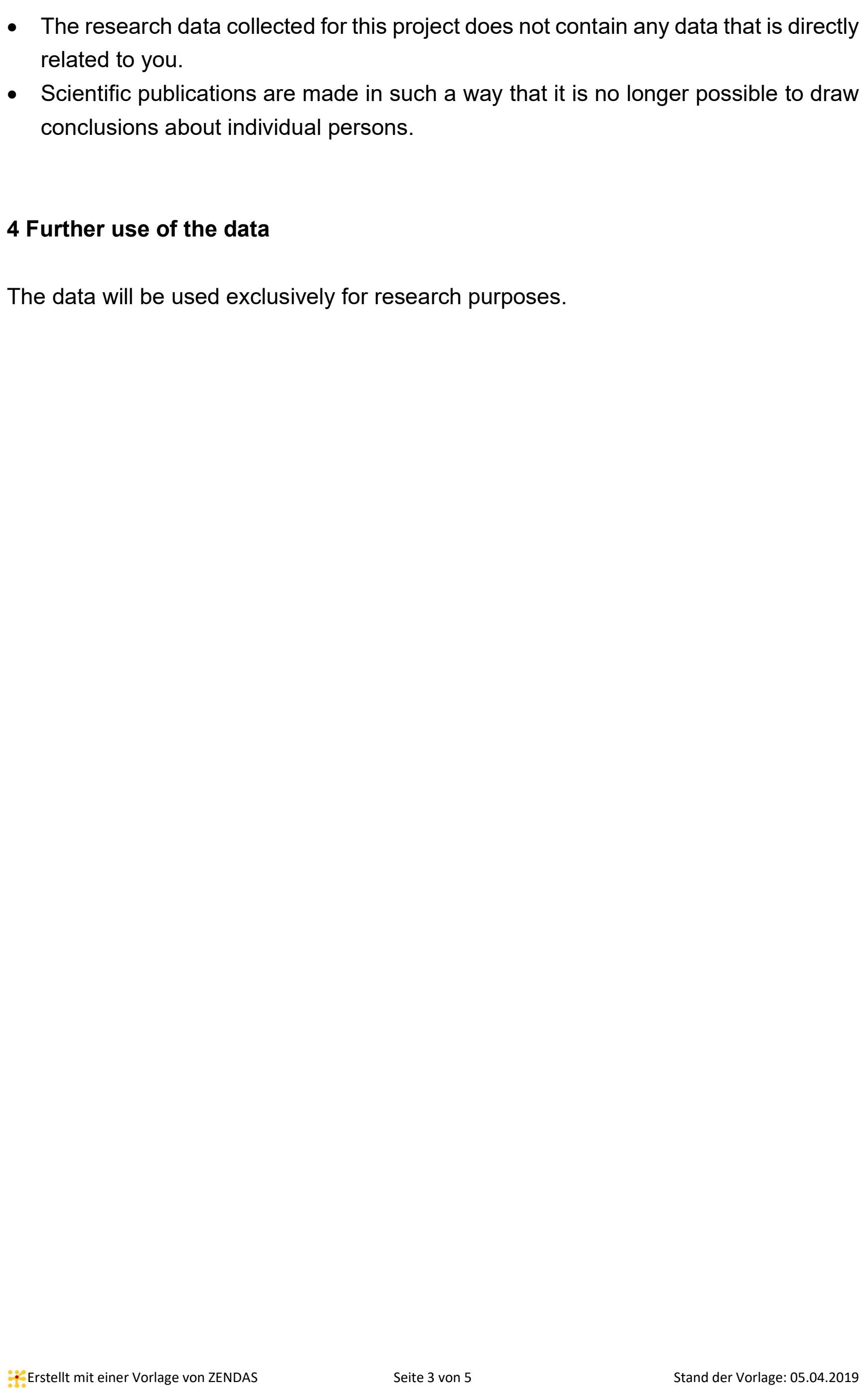
- You are at least 18 years old.

3. handling of data in the research project

- Each questionnaire and interview will be marked with a participant ID. The participant ID is assigned by the Visualization Institute of the University of Stuttgart (VISUS) and consists of letters and numbers that have no relation to your name (especially no initials or similar). All documented log data (accuracy, received reward, mouse clicks within the learning application, etc.) also get an ID. This is assigned chronologically.

Erstellt mit einer Vorlage von ZENDAS Seite 2 von 5 Stand der Vorlage: 05.04.2019

Figure A.11: Page 2 - Consent Form.



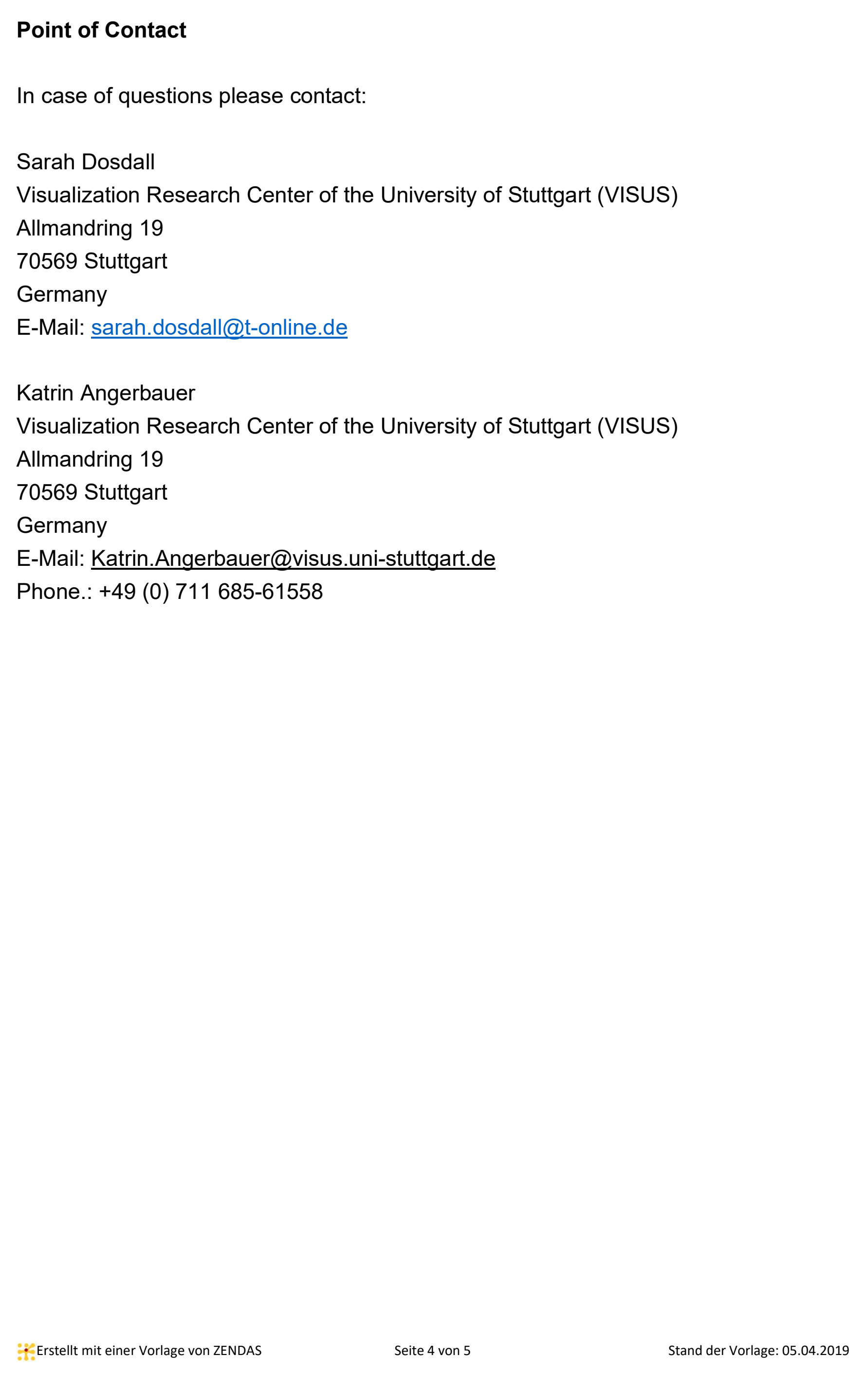
- The research data collected for this project does not contain any data that is directly related to you.
- Scientific publications are made in such a way that it is no longer possible to draw conclusions about individual persons.

4 Further use of the data

The data will be used exclusively for research purposes.

Erstellt mit einer Vorlage von ZENDAS Seite 3 von 5 Stand der Vorlage: 05.04.2019

Figure A.12: Page 3 - Consent Form.



Point of Contact

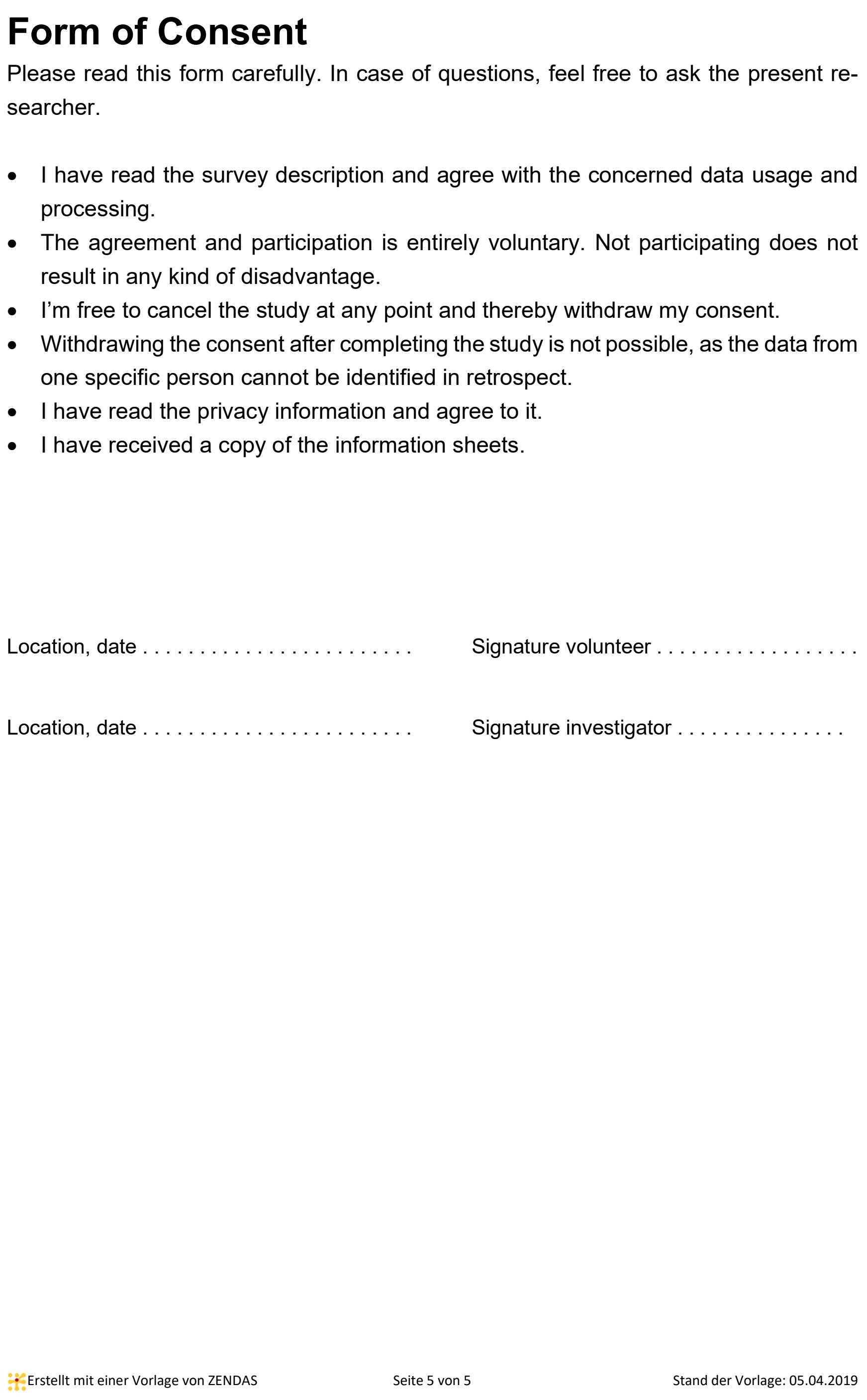
In case of questions please contact:

Sarah Dosdall
Visualization Research Center of the University of Stuttgart (VISUS)
Allmandring 19
70569 Stuttgart
Germany
E-Mail: sarah.dosdall@t-online.de

Katrin Angerbauer
Visualization Research Center of the University of Stuttgart (VISUS)
Allmandring 19
70569 Stuttgart
Germany
E-Mail: Katrin.Angerbauer@visus.uni-stuttgart.de
Phone.: +49 (0) 711 685-61558

Erstellt mit einer Vorlage von ZENDAS Seite 4 von 5 Stand der Vorlage: 05.04.2019

Figure A.13: Page 4 - Consent Form.



Form of Consent

Please read this form carefully. In case of questions, feel free to ask the present researcher.

- I have read the survey description and agree with the concerned data usage and processing.
- The agreement and participation is entirely voluntary. Not participating does not result in any kind of disadvantage.
- I'm free to cancel the study at any point and thereby withdraw my consent.
- Withdrawing the consent after completing the study is not possible, as the data from one specific person cannot be identified in retrospect.
- I have read the privacy information and agree to it.
- I have received a copy of the information sheets.

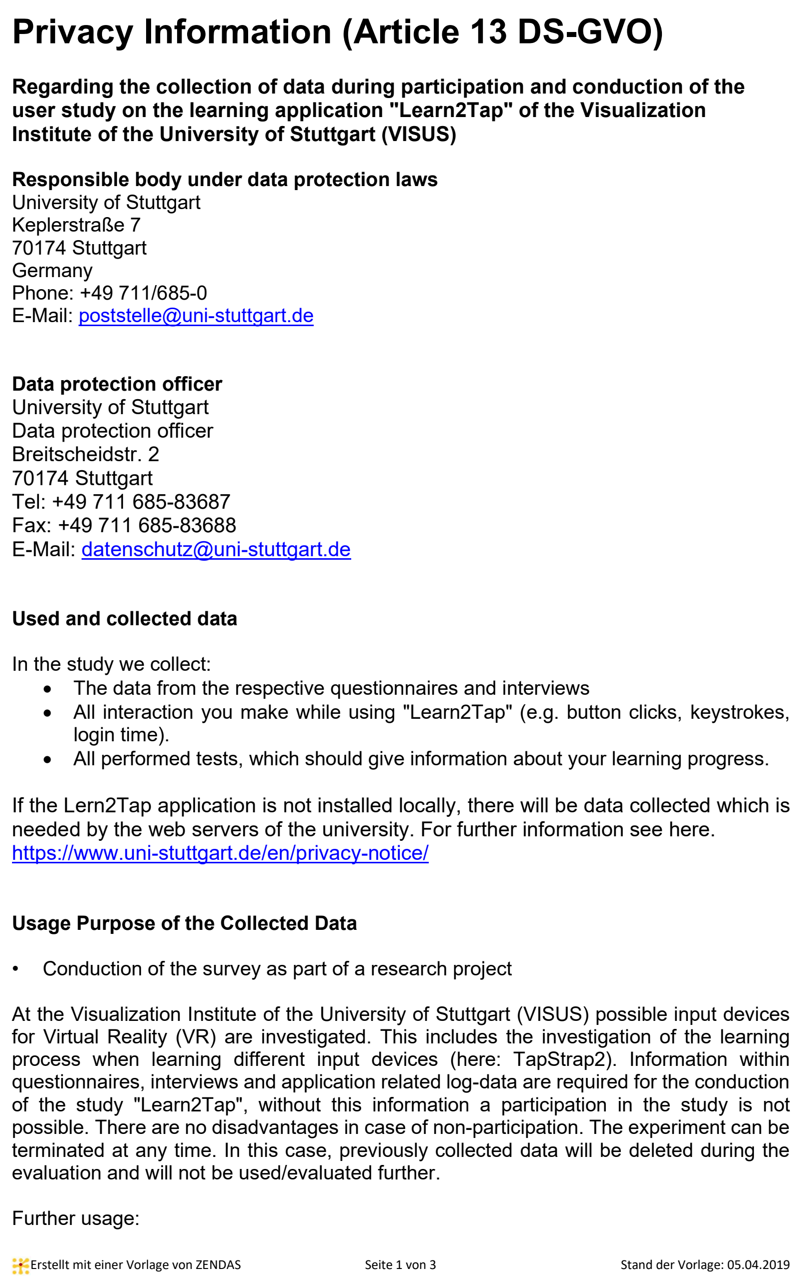
Location, date . . . .   Signature volunteer . . . .

Location, date . . . .   Signature investigator . . . .

Erstellt mit einer Vorlage von ZENDAS   Seite 5 von 5   Stand der Vorlage: 05.04.2019

Figure A.14: Page 5 - Consent Form.

A.4.2 Privacy Information



Privacy Information (Article 13 DS-GVO)

Regarding the collection of data during participation and conduction of the user study on the learning application "Learn2Tap" of the Visualization Institute of the University of Stuttgart (VISUS)

Responsible body under data protection laws
University of Stuttgart
Keplerstraße 7
70174 Stuttgart
Germany
Phone: +49 711/685-0
E-Mail: poststelle@uni-stuttgart.de

Data protection officer
University of Stuttgart
Data protection officer
Breitscheidstr. 2
70174 Stuttgart
Tel: +49 711 685-83687
Fax: +49 711 685-83688
E-Mail: datenschutz@uni-stuttgart.de

Used and collected data

In the study we collect:
- The data from the respective questionnaires and interviews
- All interaction you make while using "Learn2Tap" (e.g. button clicks, keystrokes, login time).
- All performed tests, which should give information about your learning progress.

If the Lern2Tap application is not installed locally, there will be data collected which is needed by the web servers of the university. For further information see here. https://www.uni-stuttgart.de/en/privacy-notice/

Usage Purpose of the Collected Data

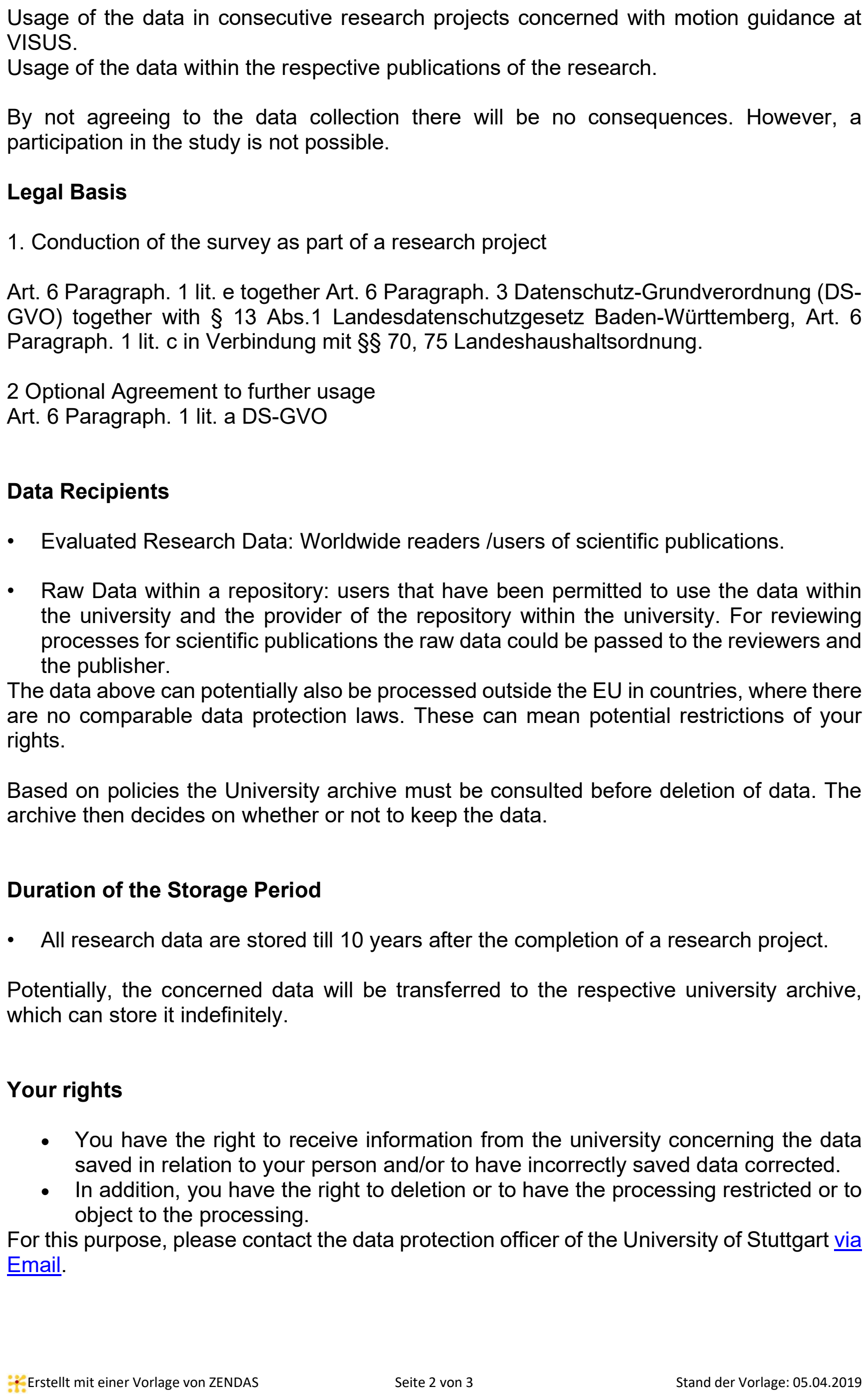
- Conduction of the survey as part of a research project

At the Visualization Institute of the University of Stuttgart (VISUS) possible input devices for Virtual Reality (VR) are investigated. This includes the investigation of the learning process when learning different input devices (here: TapStrap2). Information within questionnaires, interviews and application related log-data are required for the conduction of the study "Learn2Tap", without this information a participation in the study is not possible. There are no disadvantages in case of non-participation. The experiment can be terminated at any time. In this case, previously collected data will be deleted during the evaluation and will not be used/evaluated further.

Further usage:

Erstellt mit einer Vorlage von ZENDAS Seite 1 von 3 Stand der Vorlage: 05.04.2019

Figure A.15: Page 1 - Privacy Information.



Usage of the data in consecutive research projects concerned with motion guidance at VISUS.
Usage of the data within the respective publications of the research.

By not agreeing to the data collection there will be no consequences. However, a participation in the study is not possible.

Legal Basis

1. Conduction of the survey as part of a research project

Art. 6 Paragraph. 1 lit. e together Art. 6 Paragraph. 3 Datenschutz-Grundverordnung (DS-GVO) together with § 13 Abs.1 Landesdatenschutzgesetz Baden-Württemberg, Art. 6 Paragraph. 1 lit. c in Verbindung mit §§ 70, 75 Landeshaushaltsordnung.

2 Optional Agreement to further usage
Art. 6 Paragraph. 1 lit. a DS-GVO

Data Recipients

- Evaluated Research Data: Worldwide readers /users of scientific publications.

- Raw Data within a repository: users that have been permitted to use the data within the university and the provider of the repository within the university. For reviewing processes for scientific publications the raw data could be passed to the reviewers and the publisher.

The data above can potentially also be processed outside the EU in countries, where there are no comparable data protection laws. These can mean potential restrictions of your rights.

Based on policies the University archive must be consulted before deletion of data. The archive then decides on whether or not to keep the data.

Duration of the Storage Period

- All research data are stored till 10 years after the completion of a research project.

Potentially, the concerned data will be transferred to the respective university archive, which can store it indefinitely.

Your rights

- You have the right to receive information from the university concerning the data saved in relation to your person and/or to have incorrectly saved data corrected.
- In addition, you have the right to deletion or to have the processing restricted or to object to the processing.

For this purpose, please contact the data protection officer of the University of Stuttgart via Email.

Erstellt mit einer Vorlage von ZENDAS Seite 2 von 3 Stand der Vorlage: 05.04.2019

Figure A.16: Page 2 - Privacy Information.

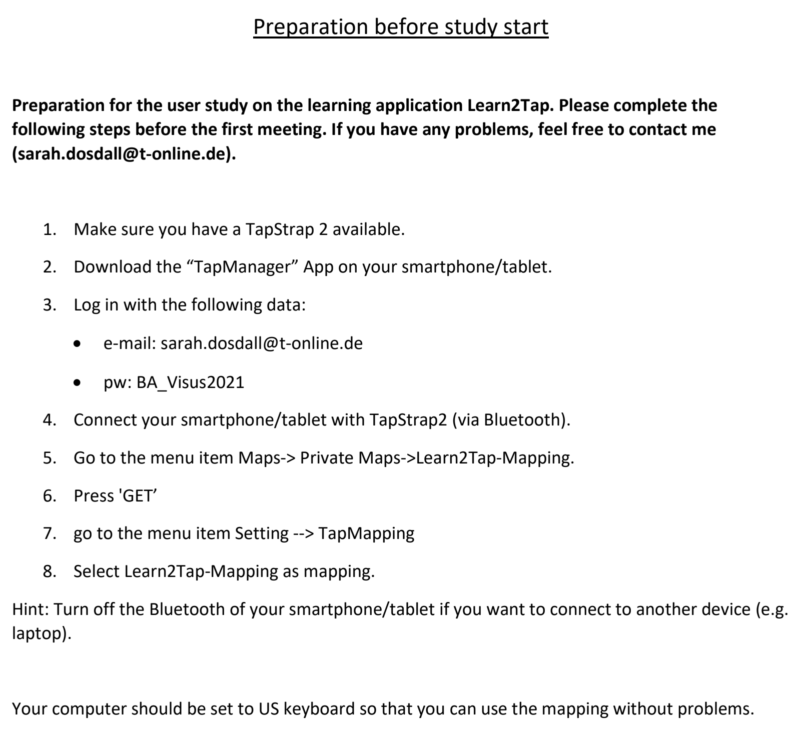
- You have the right to complain to the supervisory authority, should you be of the opinion that the processing of the personal data relating to you breaches legal regulations.

The competent supervisory authority is the State Data Protection and Freedom of Information Officer of Baden-Württemberg - Landesbeauftragte für den Datenschutz und die Informationsfreiheit Baden-Württemberg

Erstellt mit einer Vorlage von ZENDAS Seite 3 von 3 Stand der Vorlage: 05.04.2019

Figure A.17: Page 3 - Privacy Information.

A.4.3 Preparation before Study starts



Preparation before study start

Preparation for the user study on the learning application Learn2Tap. Please complete the following steps before the first meeting. If you have any problems, feel free to contact me (sarah.dosdall@t-online.de).

1. Make sure you have a TapStrap 2 available.
2. Download the "TapManager" App on your smartphone/tablet.
3. Log in with the following data:
   - e-mail: sarah.dosdall@t-online.de
   - pw: BA_Visus2021
4. Connect your smartphone/tablet with TapStrap2 (via Bluetooth).
5. Go to the menu item Maps-> Private Maps->Learn2Tap-Mapping.
6. Press 'GET'
7. go to the menu item Setting --> TapMapping
8. Select Learn2Tap-Mapping as mapping.

Hint: Turn off the Bluetooth of your smartphone/tablet if you want to connect to another device (e.g. laptop).

Your computer should be set to US keyboard so that you can use the mapping without problems.

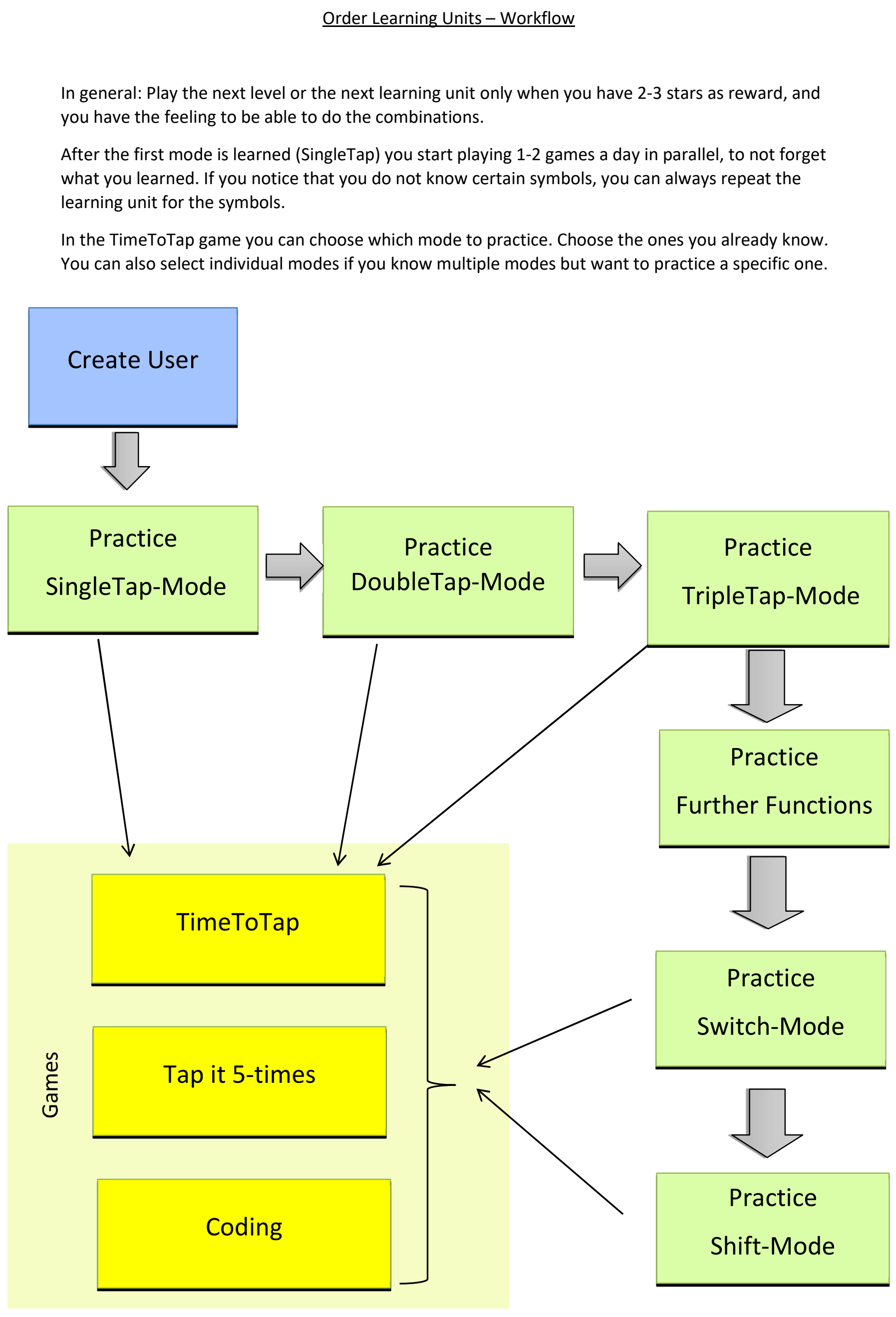
Figure A.18: Page 1 - Preparation before Study starts.

A.4.4 Demographic Data

The following questions were included in the *Demographic Questionnaire*:

1. Participant ID
2. Participant Age
3. Participant Gender
4. How experienced are you as a programmer? (1 = experienced, 5 = inexperienced)
5. What is your level of education?
6. If you have experience with the Tap Strap or a chorded keyboard, how familiar are you with it? (1 = familiar, 5 = unfamiliar)
7. If you have experience with the Tap Strap or a chorded keyboard, how long have you been using it?
8. If you have experience with the Tap Strap or a chorded keyboard, what have you used it for?

A.4.5 Order Learning Units



Order Learning Units – Workflow

In general: Play the next level or the next learning unit only when you have 2-3 stars as reward, and you have the feeling to be able to do the combinations.

After the first mode is learned (SingleTap) you start playing 1-2 games a day in parallel, to not forget what you learned. If you notice that you do not know certain symbols, you can always repeat the learning unit for the symbols.

In the TimeToTap game you can choose which mode to practice. Choose the ones you already know. You can also select individual modes if you know multiple modes but want to practice a specific one.

Figure A.19: Order Learning Units.

A.5 Questionnaires of the User Study

For the *Daily Questionnaire*, see section 5.1.

A.5.1 First Session Questionnaire

Questions about the usability:

1. Would you like to use *Learn2Tap* frequently?
2. Did you find the application unnecessarily complex?
3. Do you think the application is easy to use? Do you felt very confident using the application?
4. Would you have understood the *Learn2Tap* application without introduction video?
5. Did you find the various functions in the application were well integrated?
6. Do you think there is too much inconsistency in this application/ design?
7. Would you imagine that most people would learn to use the application very quickly?
8. Did you need to learn a lot of things before you could use *Learn2Tap*?

Interview Questions:

1. What is your first impression of *Learn2Tap*?
2. What did you like about *Learn2Tap*?
3. What could be improved about *Learn2Tap*?
4. Did you encounter any problems?
5. Do you expect that the learning application will be helpful for learning to type code with the Tap Strap?
6. Do you think you will be able to program in VR by the end of the 4th week?
7. If you hadn't gotten the sheet that says the order in which you should do the units, which one would you have started with?
8. On a scale of 1-10, how difficult did you find your exercise session today? (easy = 1, difficult= 10)
9. Please choose a feeling that summarizes your session of today:
    - Positive feelings: Amazed, Interested, Satisfied, Pleased, Optimistic, Calm, Comfortable, Pleasant, Encouraged, Wise, Surprised, Fascinated, Inquisitive
    - Negative feelings: Angry, Irritated, Annoyed, Frustrated, Depressed, Discouraged, Dissatisfied, Confused, Incapable, Tired, Indifferent, Bored, Neutral, Exhausted, Busy

A.5.2 1 Weekly Questionnaire

1. Are there any suggestions for improvement (application)?
2. Do you think the Tap Strap is a good way to program in VR?
3. If it was not for the study, would you still have motivation to continue learning the Tap Strap?
4. Do you think the stars at the end of a level are too hard to get?
5. Are 15 minutes a good length of time to practice? (too long /short)

A.5.3 2 Weekly Questionnaire

1. Are there any suggestions for improvement (application)?
2. What do you think about the game? Does it help you not to forget the combinations?
3. Do you think 15 min are too short to play a game and continue learning new combinations?

A.5.4 3 Weekly Questionnaire

1. Are there any suggestions for improvement (application)?
2. What are the reasons for pressing the 'Pause' button?
3. What would you improve about the Tap Strap?
4. Are there combinations in tapping that you would eliminate because they are too difficult to tap?
5. Do you think arranging the combinations differently would make learning easier?

A.5.5 Final Questionnaire

Interview Questions:

1. Are there any suggestions for improvement (application)?
2. How easy did you find to learn the combinations with the help of the learning application? (1 = easy, 10 = difficult)
3. Would you recommend the learning application to people who want to learn programming with the Tap Strap? Why?
4. How supportive did you find the game in terms of learning and consolidating the combinations?
5. How supportive did you find *Learn2Tap* in terms of learning the combinations?
6. Do you think that you would be able to programme in VR using the Tap Strap?
7. Would you like to programme in VR with the Tap Strap or do you prefer another input device? If so, which one?

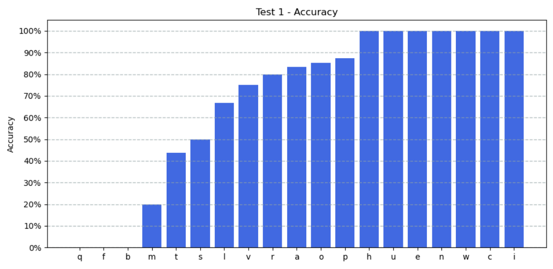
8. How long do you think it would take you to complete the remaining learning units? Would you like to complete the remaining learning units outside the study?
9. Do you think that chorded keyboard can be a viable means for typing code? What are the main benefits/drawback?

Questions about the usability:

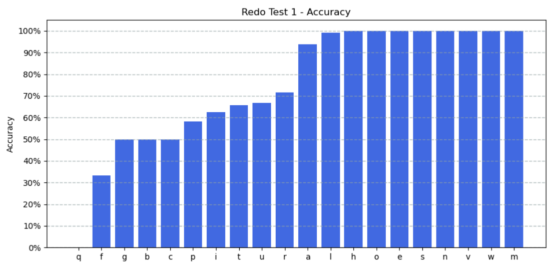
1. Would you like to use *Learn2Tap* application again if you would need to learn programming with the Tap Strap?
2. Did you find *Learn2Tap* unnecessarily complex?
3. Do you think *Learn2Tap* is easy to use? Do you felt very confident using *Learn2Tap*?
4. Did you find the various functions in *Learn2Tap* were well integrated?
5. Do you think there is too much inconsistency in *Learn2Tap*?
6. Would you imagine that most people would learn to use *Learn2Tap* very quickly?
7. Do you think the Daily Units were timed too long/short?
8. Do you become frustrated while learning due to the speed of you progress?
9. Do you think there were too many modes (Single-Tap, Double-Tap,...) that had to be learned?

A.6 User Study Diagrams

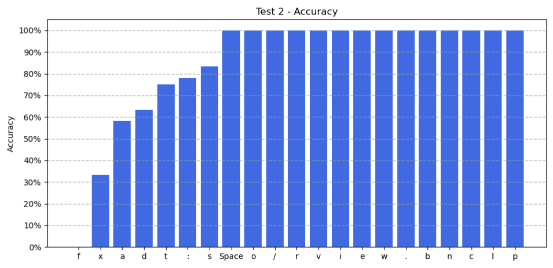
This section contains the detailed diagrams of the user study. This means that for the individual characters that occurred in the tests, the accuracy and typing speed are shown. The typing speed is only displayed for the characters that have been successfully tapped once and were not skipped each time with the 'ArrowRight' key.



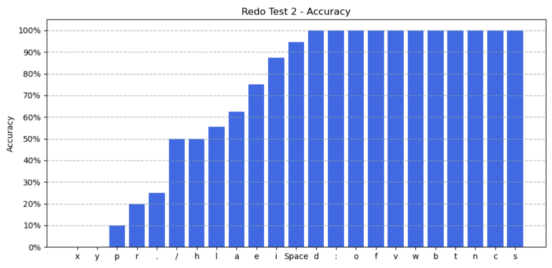
(a) Test 1.



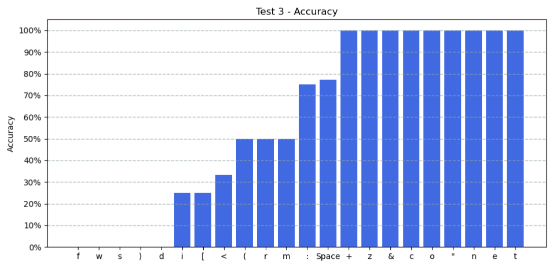
(b) Redo Test 1.



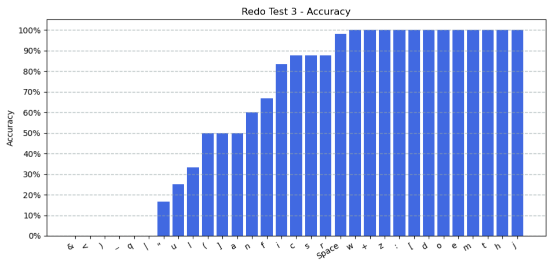
(c) Test 2.



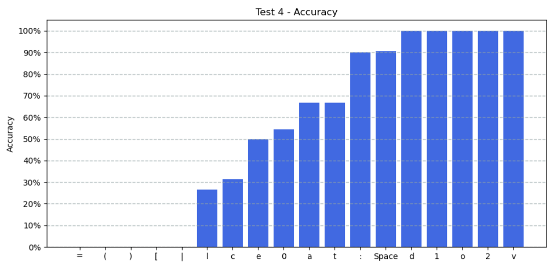
(d) Redo Test 2.



(e) Test 3.

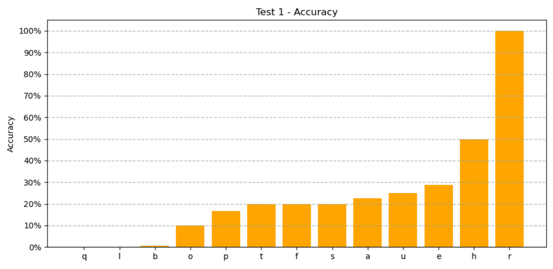


(f) Redo Test 3.

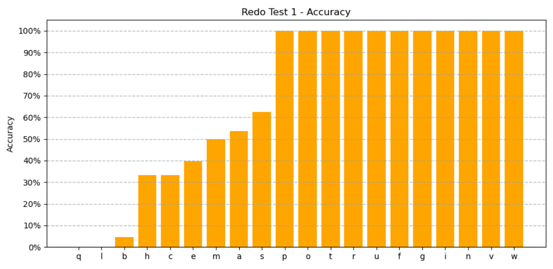


(g) Test 4.

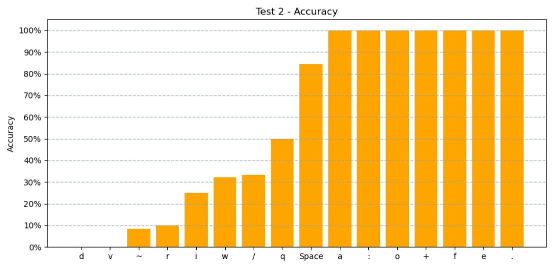
Figure A.20: The graphs above show the accuracy of u1 of the individual symbols that appeared in the tests.



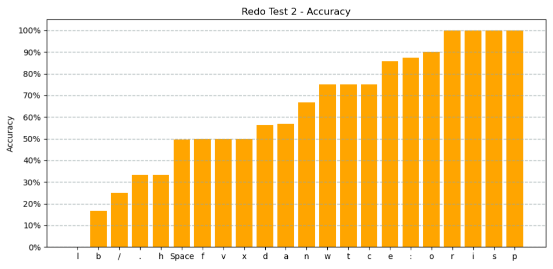
(a) Test 1.



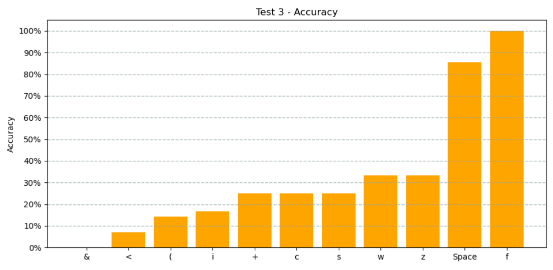
(b) Redo Test 1.



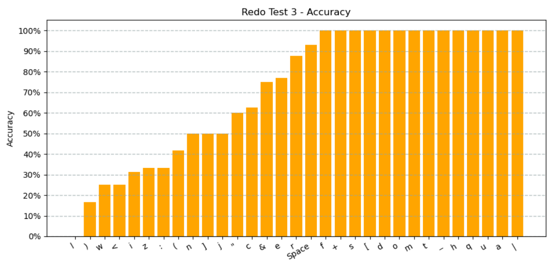
(c) Test 2.



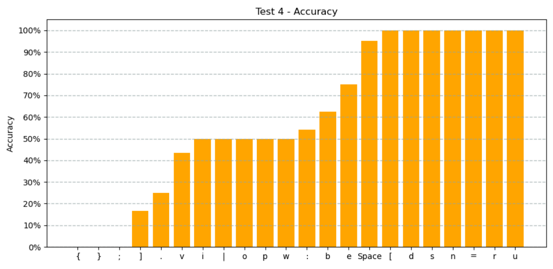
(d) Redo Test 2.



(e) Test 3.

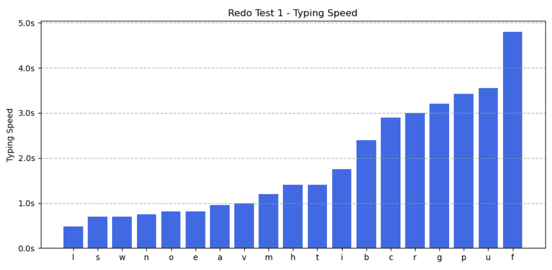


(f) Redo Test 3.

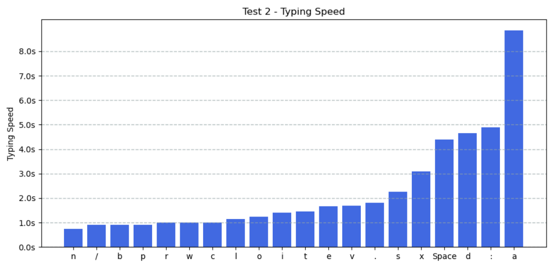


(g) Test 4.

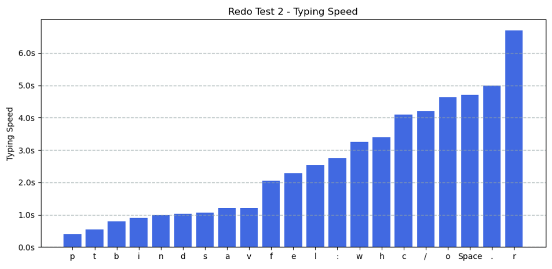
Figure A.21: The graphs above show the accuracy of u2 of the individual characters that appeared in the tests.



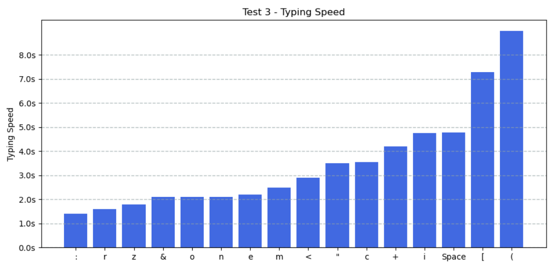
(a) Redo Test 1.



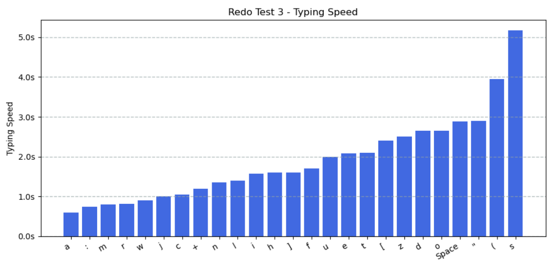
(b) Test 2.



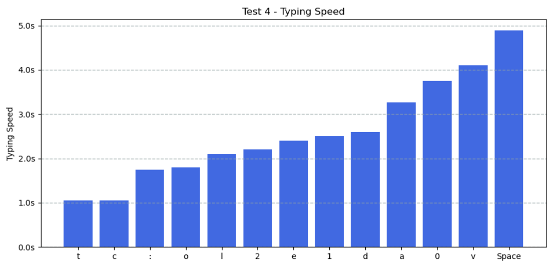
(c) Redo Test 2.



(d) Test 3.

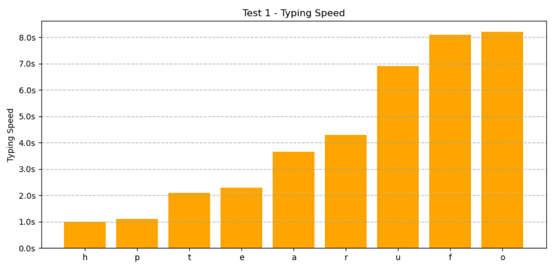


(e) Redo Test 3.

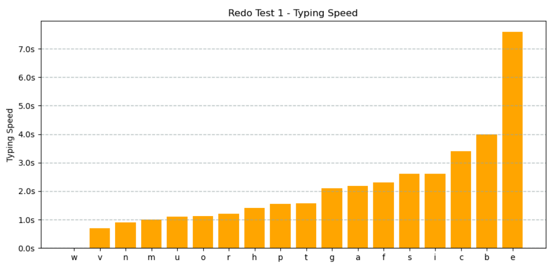


(f) Test 4.

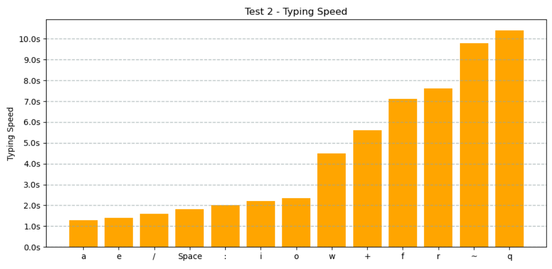
Figure A.22: The graphs above show the typing speed of u1 of the individual characters that appeared in the tests.



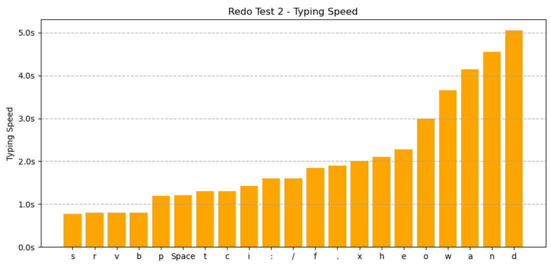
(a) Test 1.



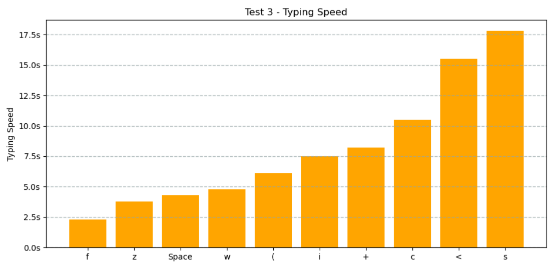
(b) Redo Test 1.



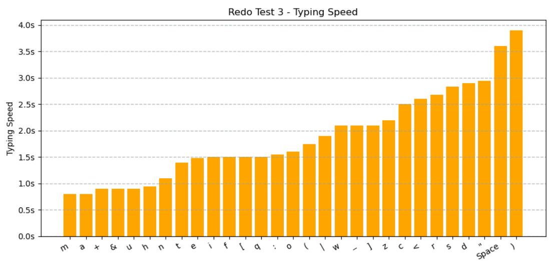
(c) Test 2.



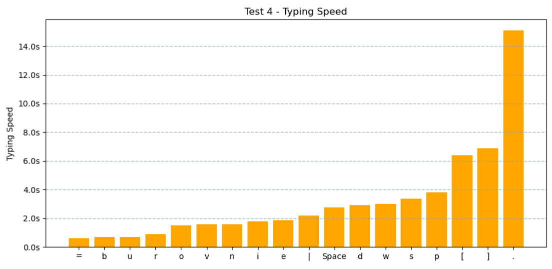
(d) Redo Test 2.



(e) Test 3.



(f) Redo Test 3.



(g) Test 4.

Figure A.23: The bar charts above show the typing speed of u2 of the individual characters that appeared in the tests.

Declaration