UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

INCA UTILS:
REFACTORING THE GRAPHICAL INTERFACES OF INCA LAB'S WEB APPLICATIONS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

FABIÁN MATÍAS JAÑA UBAL

PROFESOR GUÍA:
JÉRÉMY BARBAY

PROFESOR CO-GUÍA:
SERGIO OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:
JOSÉ MIGUEL PIQUER GARDNER
ANDRÉS ABELIUK KIMELMAN

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE:
Ingeniero Civil en Computación
POR: Fabián Matías Jaña Ubal
FECHA: 2023
PROFESOR GUÍA: Jérémy Barbay

INCA UTILS: COORDINANDO LAS INTERFACES GRÁFICAS DE LAS APLICACIONES DE INCA LAB.

InCA-Lab (https://incalab.cl/) es un laboratorio de investigación en Interacción Computacional Animal (ACI) que desarrolla aplicaciones web utilizables por Otros Animales No Humanos y/o sus guardianes. Las primeras tres aplicaciones, con interfaces gráficas distintas, dan lugar a algunos primeros paradigmas centrales. ¿Pueden estos paradigmas incorporarse en una única biblioteca de interfaz gráfica con el fin de uniformar las interfaces gráficas de las aplicaciones presentes y futuras? Diseñamos una biblioteca de este tipo, llamada InCA-Utils, y validamos su usabilidad tanto con sujetos de otras especies que no son humanos como con desarrolladores humanos que desarrollan nuevas aplicaciones para InCA-Lab.

ABSTRACT FOR MEMORIA
TO APPLY TO DEGREE OF:
Ingeniero Civil en Computación
BY: Fabián Matías Jaña Ubal
DATE: 2023
PROFESSOR: Jérémy Barbay

INCA UTILS: REFACTORING THE GRAPHICAL INTERFACES OF INCA LAB'S WEB APPLICATIONS.

InCA-Lab (https://incalab.cl/) is a research laboratory in Animal Computer Interaction (ACI) developing web applications usable by Other Animals Than Humans and/or their guardians. The first three applications, with distinct graphical interfaces, yield some first central paradigms. Can those paradigms be embedded in a unique graphic interface library in order to uniforms the graphical interfaces of present and future applications? We designed such a library, called *InCA-Utils*, and validated its usability with both subjects from other species than humans and with human developers developing new applications for InCA-Lab.

A special acknowledgement to Jérémy who motivated and guided me throughout my thesis. With his help, I was able to co-author not just one, but two research papers, something I could not have imagined. It was a pleasure working with him during this process.

Table of Content

Chapter 1

Introduction

InCA (Interacciones Computacionales para Animales) is a group that develops applications for non-human animals in digital devices, in order to support the study of their cognitive and sensory capabilities in a playful and non-coercive way. Its goals include, in the short term, improving human understanding of how animals communicate among their own species and with animals of different species, in particular between humans and Other Animals Than Humans (OATHs), using technological resources. In the long term, anticipating climate change in various regions of the planet, the project aims to design a new synergistic relationship between humans and non-human animals, not only by improving communication between them, but also by suggesting new forms of communication that can be transmitted to future generations in a semi-automated way. At present, different applications have been incorporated by the group of researchers, which aim to investigate different capabilities of the OATHs involved.

Although the applications have several elements in common among them, these elements have not been reused; rather, each implementer tends to create its own solutions. This brings different problems to the table. First, it generates a disparity in terms of common components design, although this gives more variety to the styles of the different applications, this could negatively affect the learning process of OATHs when confronted with new applications. Second, there is the constant reinvention of elements that have already been created; a developer may have created a solution that works perfectly for a large number of scenarios, but there is no an easy way to share and reuse this solution, in addition to not having a space to receive proposals for improvements to it.

Two hypotheses are put forward for this problem. The non-human subjects studied respond better to interfaces with which they are already accustomed to interact, and the reuse of already created components by developers alleviates their implementation work.

We designed a library *InCA-Utils* to centralize recurrent design issues in various web applications developed by InCA Labs, from a set of icons for recurrent actions such as starting the time for an activity (*Ready* button), exiting an activity (*Exit* Button), button protected such that subjects are unlikely to press them by mistake (*Fullscreen* and *Settings* buttons), the selection of colors and sounds (to adapt to each subject's range of perceptions, and to keep it uniform from one application to another one), the connection to the Firebase database

for the collection of usage logs.

We validated the usability of such library 1) with subjects from other species than humans by developing the variant *InCA-CompareFast* of the existing application *InCA-CompareFast*, which integrated the new graphical interface in addition to new features such as a prototype of engine to automatically adapt the difficulty, and the addition of time statistics to the data recorded by the application; and then 2) with human developers integrating such library in the design of their own application (e.g. *InCA-WhatIsMore*, *InCA-PopUp* and *InCA-ComBoard*), receiving feedback on the usefulness of the library in term of time savings and easiness of integration.

After describing in more details existing solutions related to our topic in Chapter 1, we delve into the design of the library in the Chapter 3. In Chapter 4 we will discuss the implementation details for each functionality of the library, the Chapter 5 will cover most of the validations made by users and developers using the library. Finally we will conclude on Chapter 6, outlining potential future work.

Chapter 2

State of the Art

Each section will first give an overview of common features that are desirable when developing an application that seeks to measure cognitive abilities of OATHs (other animals than humans), it will then allude to problems faced when wanting to implement these features and refer to some solutions currently implemented by developers of InCA Labs.

2.1 The non-human animal as the main user

The applications developed by InCA Labs are focused on OATHs being the ones who interact directly with the application, they are the ones who decide what kind of game they are going to play and decide when to stop playing. The interaction of the guardian with the application takes a back seat.

Let's take *InCA-WhatIsMore* as an example, when opening the application the subject is directly presented with the available game modes, from which he can choose which one to play (See Figure 2.1). Once inside the game the subject can at any time decide to exit the game mode by pressing the exit button (See Figure 2.2).

This feature introduces the following requirement to applications:

Protection Mechanisms

As mentioned above, an important focus of these applications is the subject's feeling of "agency", i.e. the OATH can control many aspects of the application and play without the intervention of its guardian. However, it is necessary to have exclusive views for humans, either to change configurations or do some guardian specific action. For this purpose, some protection mechanisms are introduced to protect these views.

- In *InCA-WhatIsMore* the protection mechanism is a long press button. Certainly with this protection OATHs using this application were not able to enter this view, but it

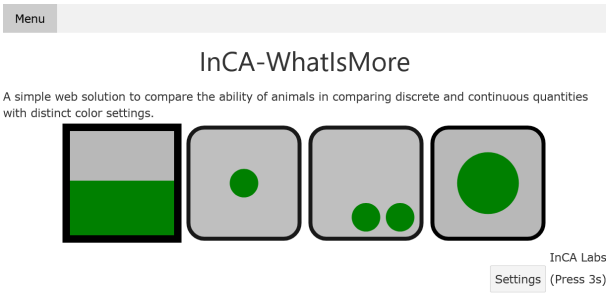

Figure 2.1: WhatIsMore main menu: Each of the four tiles represent a game mode, OATHs can decide which mode to play just picking one.

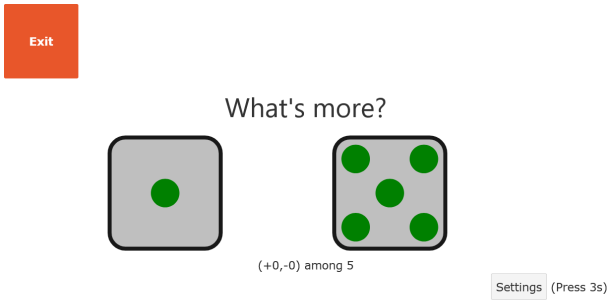

Figure 2.2: WhatIsMore ingame view: Here we see that the user is given an option to leave the game whenever he wants.

introduces problems like having to make explicit that the button needs a long press to be activated (using a label) and not having a visual feedback to indicate the time of pressing (See Figure 2.3).

- In a prototype of *InCA-CompareFast* the same type of solution is proposed, but trying to solve the problems already described in *InCA-WhatIsMore*, the result was a button that fills up according to how long it is held down. This solves the second problem of the *InCA-WhatIsMore* button, but still needs an indicator that it is a long-press button, since it needs to be held down for a minimum time to activate the visual feedback and when pressed normally there is no indication that it is working (See Figure 2.4).

2.2 Species-Specific Considerations

Different animal species have unique sensory capabilities and cognitive abilities. It is crucial to understand the target species' natural behaviors and preferences when designing the application. The colors displayed by digital displays and the sound frequencies played by devices are optimized for the majority of humans. It is not always clear how much and which colours and sound can be seen and heard by individual of each species [3]. Applications should offer options to handle the set of color used to adapt to the OATH senses.

In the cases of *InCA-WhatIsMore* and *InCA-ClickInOrder*, both offer similar configurations to change the color of tiles, but lacks on options to change other elements of the interface, like buttons and background (see Figure 2.5).

2.3 Experimental Biases

The history of Comparative Psychology has been prone with arguments about the validity of methodologies and results: Pepperberg [11] describes various such tensions between

Figure 2.3: *InCA-WhatIsMore* protection button: Longpress button without feedback. Uses a label to indicate that it's a longpress button.

Figure 2.4: Protection button from an early version of *InCA-CompareFast*: The image shows the button partially activated. The semi-transparent overlay depicts the button in an intermediate state of activation, providing visual feedback of a prolonged press.

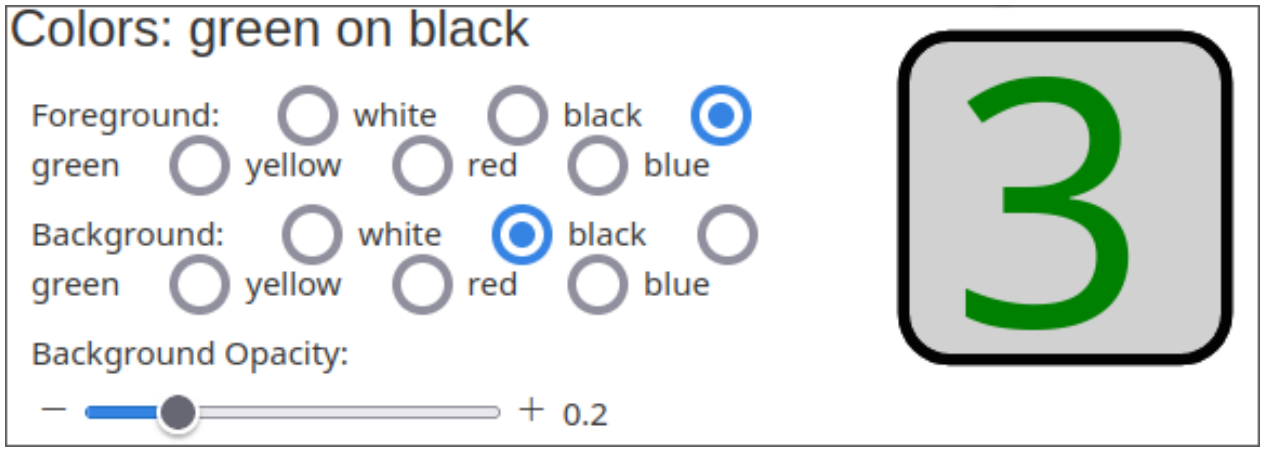

Figure 2.5: *InCA-WhatIsMore* and *InCA-ClickInOrder* simple color customization. The figure shows the options configurations to change the background color, foreground color and background opacity of a tile.

researchers about the psychology of animals, with some accusing other researchers in the field to be "*liars, cheats and frauds*", and she highlights how sign language researchers were accused of "*cuing their apes by ostensive signals*" and of "*consistently over-interpreting the animals' signs*". We explore here two issues relevant to the experimentation protocol used by InCA applications, namely *selective reporting bias* (Section 2.3.1) and the *"Clever Hans" effect* (Section 2.3.2).

2.3.1 Selective Reporting Bias

Selection biases occur in a survey or experimental data when the selection of data points is not sufficiently random to draw a general conclusion. Selective reporting biases are a specific form of selection bias whereby only interesting or relevant examples are cited. Cognitive skills can be particularly hard to study in nonhumans, requiring unconventional approaches which often present an increased risk of such biases. For example, an experimenter who would present a subject repeatedly with the same exercise could be tempted to omit or exclude bad performances (eventually attributing them to a "bad mood" of the subject, which remains a real possibility) and report only on good performances, creating a biased representation of the abilities of the subject, leading to a selective reporting bias.

2.3.2 "Clever Hans" effect

Among such methodological issues resulting in experimental biases, the most iconic one might be the case of the eponymous horse nicknamed "Clever Hans" which appeared to be able to perform simple intellectual tasks, but in reality relied on involuntary cues not only given by their human handler, but also by a variety of human experimenters. It is possible to avoid the confusion between a subject's ability to read cues from the experimenter from its ability to answer the tests presented to them by such an experimenter. The principle is quite simple: make sure that the experimenter does not know the test, by having a third party out of reach from the subject's reading to prepare the test. Whereas such experimental setup was historically referred to as a "Blind Setup" or a "Blinded Setup", we follow the recommendations of Moris et al. [1] and prefer the term of "masked" to the term "blind" when describing the temporary and purposeful restricted access of the experimenter to the testing information.

To mitigate the aforementioned issues, applications developed by Inca Labs, including *InCA-WhatIsMore*, offer an audio feedback system. Such a system provides an impartial and consistent way of conveying feedback to the OATHs guardians on "masked setups".

In its earlier version, *InCA-WhatIsMore* used recorded audio files in mp3 format. This solution, while functional, was not very scalable or configurable. In subsequent iterations, I revised this system and implemented a more configurable version. This new system allows users to modify the audio scripts and change the language, enhancing its adaptability to various experimental setups. However, there are other desirable configurations that have yet to be implemented, such as the ability to adjust the volume, pitch, and speed of the audio feedback.

Chapter 3

Design

In this chapter, we will explore the core concepts related to the library and its functionalities. We will delve into the set of functionalities provided by the library, explaining the design of each components and APIs of it. In the last section we will discuss the design approach that prioritizes an enhanced developer experience.

3.1 TrainerButton

The design of the TrainerButton component focuses on accessibility and ease of use, especifically for OATHs' guardians. It incorporates features such as a clear visual indication of the button state, non-linear progression, and customizability to adapt to various application contexts.

3.1.1 Protected Button

The TrainerButton is designed as a protected button. This design choice ensures that only the guardians of OATHs can trigger it, avoiding unintended triggers. The implementation of a long-press action, coupled with clear visual and haptic feedback, ensures that any action tied to the button is initiated with a full understanding of its function and with clear intent. The Figure 3.1 shows the TrainerButton in its idle state, and Figure 3.2 displays the button during a long-press action, where the fill effect and shadowing are apparent.

3.1.2 Button Structure and Visual Indication

The *TrainerButton* adopts a low-profile design, keeping a clean and non-distractive look to avoid attracting the attention of OATHS. The button is structured as a button element that accommodates an SVG icon (or any other visual representation) within it, allowing for a customizable appearance. The button also includes a label, which can be set as per the

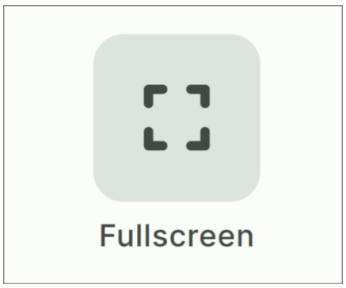

Figure 3.1: An illustration of a protected button, which triggers a fullscreen toggle action. The button prominently displays an 'expand' icon in the center and is labeled 'Fullscreen' beneath.

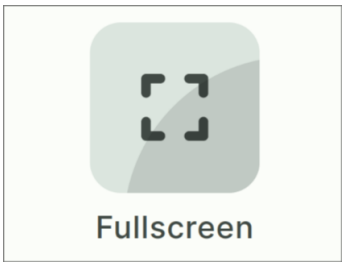

Figure 3.2: The protected button 'Fullscreen', now showing a rounded shadow at the bottom right corner. This shadow represents the progress of the long-press action required to trigger the button's function.

specific context of its use.

The button includes visual feedback that accurately represents its state. When the button is pressed, a fill effect takes place to indicate that the button press is registered. The design employs a semi-transparent *div* that overlays the button and creates a shadowing effect.

3.1.3 Non-linear Progression

The TrainerButton employs a non-linear progression for the long-press action. This design decision aids the guardian to realize that the button is of the long-press type right at the first contact. If a linear function was used, a regular touch would not provide any significant visual effect. By opting for a non-linear progression, even a short touch interaction provides a noticeable fill effect, making the functionality of the button apparent.

The non-linear progression is implemented using a cubic bezier function. The figure 3.3 illustrates the specific function utilized, with the curve demonstrating the progression of the fill effect over time.

3.1.4 Haptic Feedback

When the action is successfully triggered (after a complete long-press), the *TrainerButton* provides haptic feedback. This tactile response further enhances the button's accessibility by providing immediate confirmation of the action trigger.

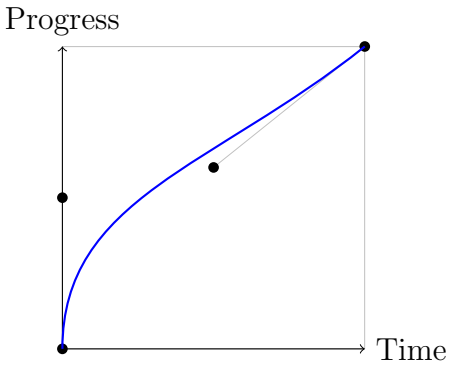

Figure 3.3: A cubic Bezier function represented by four points (0, 0.5, 0.5, 0.6), illustrating an eased transition timing for the fill animation in the TrainerButton component. The curve indicates a rapid initial acceleration, followed by a gradual decrease in speed, thus enhancing the visual feedback for the user.

3.2 ActionButton

The ActionButton is a user interface component designed explicitly for the interaction of the OATHs within the application. The design philosophy of the ActionButton focuses on enhancing subjects' sense of 'agency,' reinforcing the idea that they are in control of their actions within the application.

3.2.1 Design Elements

The ActionButton incorporates a square button housing an icon that intuitively represents its functionality. The decision to use simple yet distinct icons (a play symbol and a left pointing arrow) aids users in quickly identifying the purpose of the button,

3.2.2 Color Cues

A significant element in the design of the ActionButton is the strategic use of colors to impart visual cues about the button's function. The application utilizes vibrant, unambiguous colors that cater to the OATHs' perception, lending to a more intuitive experience.

When the button is in 'ready' mode, representing a trigger or initiation action, it features a vibrant yellow color (see Figure 3.4). In contrast, when is in 'exit' mode, indicating an action to cancel or move back, the button features a bright red color (see Figure 3.5).

Figure 3.4: A image representing a 'ready' ActionButton. It shows a simple square button housing a 'play' icon and using a yellow color scheme.

Figure 3.5: A image representing a 'stop' ActionButton. It shows a simple square button housing a 'left arrow' icon and using a red color scheme.

3.3 Tile

This component is specifically designed to gauge the ability of OATHs to differentiate between different quantities, both discrete and continuous. For discrete values, the component may represent quantities as a number of dots on a grid or the number of dots on a dice face. For continuous quantities, it could use visual cues such as the degree to which a square is filled or the size of a circle.

The component introduces a serie of representation modes:

- Dice: Dots are displayed in a grid, emulating the different faces of a die. This mode mimics the common sight of a dice roll, making it familiar and easy to understand.
- Rect: This is a continuous representation mode that simulates the filling percentage of a square. The more the square is filled, the higher the represented quantity.
- Disc: This is a continuous representation mode that measures the size of a circle. The larger the circle, the higher the represented quantity.
- Heap: This representation utilizes a grid of points that fill from the bottom to the top, layer by layer. The total number of points across the grid indicates the represented quantity.
- Stack: In this grid-type representation, points accumulate from the bottom upwards, forming tower-like stacks. The total number of points across the grid indicates the represented quantity.
- Grid: This representation displays points in a grid, distributed randomly. The total number of points across the grid indicates the represented quantity.
- Donut: This representation counts the number of divisions within a donut-shaped figure. The number of sections indicates the represented quantity, allowing for a visually intuitive understanding of different quantities.

3.4 Speech API

Reflecting upon the historical challenges of biases in Comparative Psychology, it's clear that audio feedback plays a pivotal role in mitigating these issues. Specifically, it helps counter 'selective reporting' bias (see Section 2.3.1), where non-random selection of data could lead to inaccurate results, and the 'Clever Hans' effect (see Section 2.3.2), where the experimenter's inadvertent cues influence an animal's response. Through the implementation of a 'masked' setup, where the experimenter is blind to the test specifics, these biases can be significantly reduced.

To meet this need for audio feedback, we propose a customizable text-to-speech system in our library. This system allows for the adjustment of language, speed, volume, and pitch according to each user's requirements. The availability of such a flexible tool not only helps maintain the reliability and objectivity of experimental outcomes but also enhances the overall user experience.

3.5 FullScreen API

This feature is crucial when using the library with OATHs on mobile devices. Without it, animals tend to unintentionally activate the phone's navigation buttons, the status bar, or the browser's navigation bar. This fullscreen functionality ensures an unobstructed and focused interaction for the animals, minimizing unwanted disruptions.

3.6 Theme API

It's essential to provide various color and size configurations to cater to the unique abilities of each OATH. Not only does this customization allow for optimal usability, but it also presents a valuable opportunity for study. By exploring different color ranges, we can learn more about the OATHs' color perception abilities, potentially surpassing what is known about human capabilities.

The Theme API is designed with customization in mind, offering extensive control over the color scheme of the entire application.

3.6.1 Predefined Colors

The API provides four predefined themes, delivering a convenient starting point for users who prefer a quick setup. Each theme has been carefully crafted, taking into account accessibility standards, ensuring a balanced and user-friendly color scheme right out of the box.

3.6.2 Generation of New Schemes

To accommodate the need for a more personalized color scheme, the API is designed to generate new themes from a single base color. This feature enables a high level of customization while maintaining consistency and harmony throughout the application.

3.6.3 Detailed Customization

For users who desire an even more personalized aesthetic, the Theme API offers detailed customization options. Users can independently modify the color of the background, text, and specific components such as buttons and tiles. This level of detail provides the flexibility to tailor the application to meet unique aesthetic preferences or specific accessibility requirements.

3.6.4 ThemePicker Component

To effectively demonstrate the potential of the Theme API and facilitate its integration, a special component named *ThemePicker* was developed as part of the *InCA-CompareFast* project. This component acts as a showcase of all the functionalities that the Theme API can provide. (see Figure 3.6)

However, *ThemePicker* serves only as a template or a starting point. It's designed to be adapted and extended by developers to suit their specific application requirements. Its primary purpose is to give developers a practical and tangible example of how to implement the Theme API.

3.7 Development Experience

InCA-Utils is designed to offer a seamless and enjoyable development experience, prioritizing key aspects such as documentation, package distribution, IntelliSense, and versioning. These considerations aim to empower developers and facilitate their integration of the library into their projects. Let's explore how the library enhances the development experience.

3.7.1 Comprehensive Documentation

The library provides extensive and well-structured documentation. It includes detailed explanations, code examples, and tutorials to guide developers through the library's functionalities. The documentation serves as a valuable resource, enabling developers to quickly understand and effectively utilize the library's features.

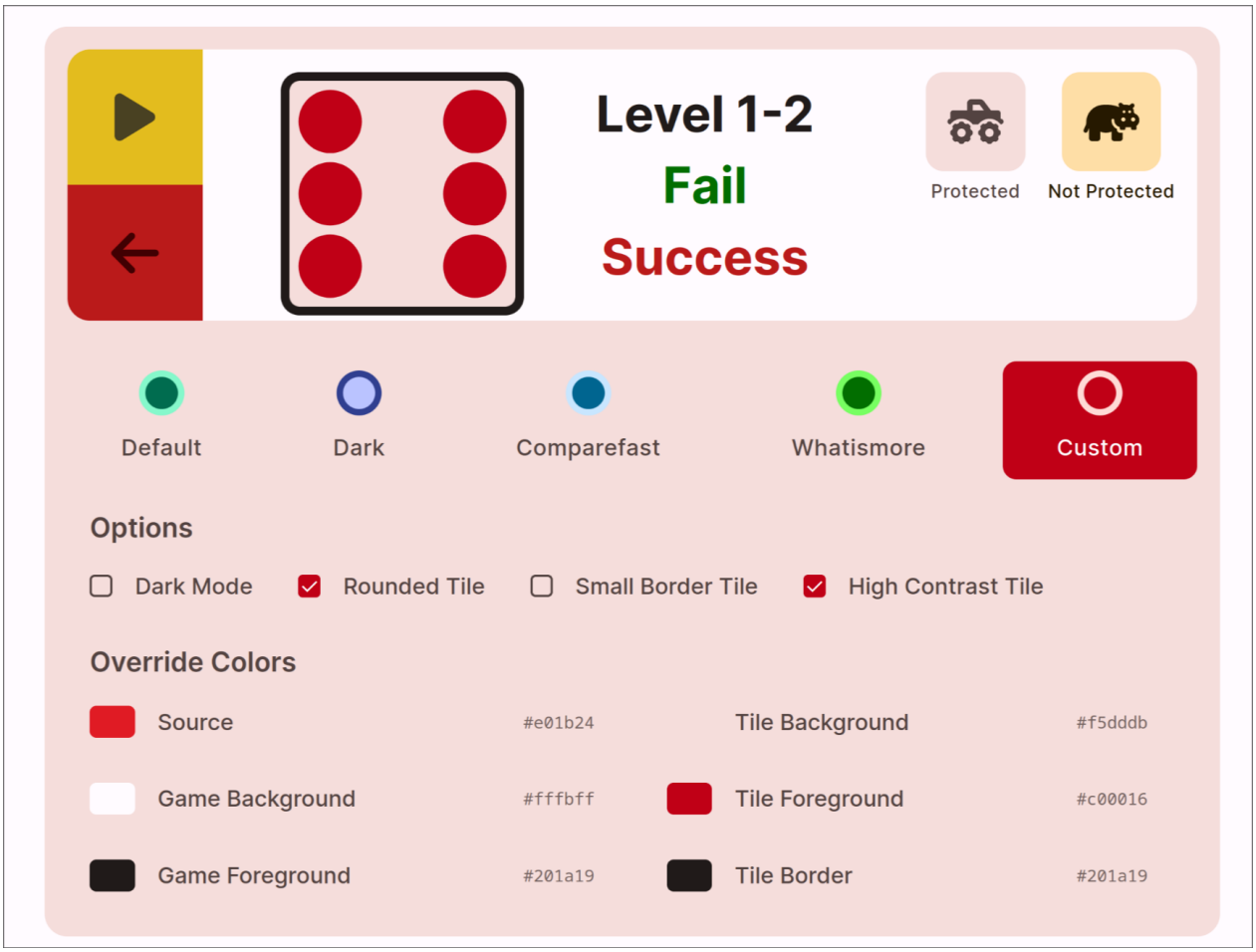

Figure 3.6: The ThemePicker component in *InCA-CompareFast*, showing the "Custom" tab. This tab allows users to define their own theme, each color option represents a specific aspect of the user interface. The component displays a preview on the top to show how each color affects all the components of the application.

3.7.2 Package Distribution and Management

The library is distributed as a well-packaged and easily installable package through a popular package manager. This simplifies the integration process, allowing developers to effortlessly include the library as a dependency in their projects. The package management approach ensures version control, simplifies updates, and facilitates dependency management.

3.7.3 IntelliSense Support

The library includes IntelliSense support, providing real-time code completion, parameter information, and function signatures within integrated development environments (IDEs). This feature significantly improves developer productivity by offering contextual suggestions and reducing the time spent on manual lookups and referencing the external documentation.

3.7.4 Versioning

The library adheres to a versioning system that follows best practices. Clear release notes accompany each version, outlining new features, bug fixes, and any potential breaking changes.

Chapter 4

Implementation

This chapter provides a detailed overview of the implementation process for InCA Utils, detailing the various features and how they were realized within the library.

4.1 Environment

Most of InCA Labs' implementations primarily use Svelte and SvelteKit. Following this line, the decision to develop the library was made with this in mind. The development process utilized a robust web stack, including Svelte as the primary framework (compatible with versions 3 and 4, as well as with SvelteKit), npm as the package manager, and several browser APIs like CSSStyleSheet, Web Speech API, and Fullscreen API. Furthermore, we incorporated TypeScript as the main programming language to enable static type checking and enhance the library's maintainability. Git was also used extensively for version control, facilitating efficient and organized collaboration amongst the development team.

4.1.1 Svelte & SvelteKit

Svelte is a *JavaScript* framework that facilitates the creation of interactive user interfaces in the web browser. *SvelteKit* is a framework built on top of *Svelte*, providing additional functionalities such as routing and server-side rendering (SSR).

InCA-Utils supports both of these frameworks, spanning from version 3.50 to the newly released version 4, which came out in July of this year.

4.1.2 TypeScript

Typescript is a statically typed superset of JavaScript that adds optional types, classes, and modules to the language, among other features.

This library employs TypeScript to improve maintainability and prevent potential type-related errors during development. Also gives better tooling for developers using the library like autocompletion and type checking.

While TypeScript has been employed in the development of the library, it's not required for developers who are using this. The library outputs standard JavaScript code alongside TypeScript declaration files (.d.ts), this way developers can still benefit from suggestions and autocomplete in many code editors regardless of whether they use *TypeScript* or standard *JavaScript*.

4.1.3 Node.js & npm

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It is used to execute JavaScript on the server-side. npm (Node Package Manager), on the other hand, is a package manager for JavaScript and the world's largest software registry. It is bundled with Node.js by default and facilitates the sharing and borrowing of packages from other developers.

For InCA-Utils, npm serves as the primary distribution channel. The library is packaged and published to npm, allowing developers to download and include it in their projects easily. To use the library, developers only need to run the command *npm install inca-utils*, and npm will fetch the library from its registry and install it in their project.

4.1.4 GitLab

GitLab is a web-based DevOps tool that provides a Git-repository manager. It was used to track changes in the source code during software development.

4.2 TrainerButton Component

The TrainerButton implementation is crafted using Svelte, taking advantage of its reactivity model and event dispatching capabilities.

4.2.1 HTML Structure

The TrainerButton is made up of a button element containing a slot for any graphical representation (like an SVG icon) and a conditionally rendered label. This HTML structure offers versatility in designing the button with various icons and textual labels.

4.2.2 Svelte's Reactivity

Svelte's reactivity model is employed to maintain the button's state, particularly whether the button is being held and the duration of the press. The *holding* variable is used to control the state and visual response of the button.

4.2.3 Event Handling

The TrainerButton utilizes mouse and touch events (*mousedown*, *mouseup*, *mouseleave*, *touchstart*, *touchend*, *touchcancel*) to detect and manage user interaction. The *start* function is triggered by *mousedown* and *touchstart* events, setting the *holding* state to *true* and a adding a timer to dispatch the 'click' action. The *cancel* function is activated by *mouseup*, *mouseleave*, *touchend*, and *touchcancel* events to reset the *holding* state and cancel the timer.

4.2.4 Button Animation

The animation of the TrainerButton involves both a filling visual (when the button is long-pressed) and a shadowing effect to simulate a physical button press. This animation is realized through CSS transitions.

The button filling is achieved through a non-linear progression using a CSS cubic-bezier function.

The shadowing effect is a result of translating a semi-transparent *div* (overlaid on the button) when the *holding* state is true. This *div*, which is not visible by default (being outside of the button area when holding is false), moves on top of the button when the *holding* state is true, creating the effect of a button being pressed.

These animations are associated with the holding state of the button, and the duration is set based on the *longpressTime* variable.

4.2.5 Haptic Feedback

The implementation of haptic feedback is realized using the Web Vibration API [13]. When the 'click' event is dispatched, a short vibration is triggered, giving the user tactile feedback of a successful long press action.

4.3 ActionButton Component

This *Svelte* component represents the implementation of the *ActionButton* introduced on Section 3.2.

4.3.1 HTML Structure and Event Handling

The HTML structure of the ActionButton is a single button element, with the icon determined by the mode state. The button listens for a click event, which triggers an action based on the current mode.

4.3.2 Styling and Mode-dependent UI

The button's styling is determined by the current mode. We have two classes, 'ready' and 'exit', and these classes are toggled on the button element based on the mode state. This makes it possible to have different styles for the button based on whether it is in 'ready' or 'exit' mode.

4.4 Tile Component

Implementing the design requirements detailed in Section 3.3 was largely achieved through extensive use of SVG, a powerful tool for creating scalable, high-quality graphical elements.

The Tile component consists of several distinct representation modes, each with its unique layout and graphical representation. For modes that visualize quantities as dots on a grid - namely Dice, Heap, Stack, and Grid - a common positioning algorithm was developed. This algorithm accepts the current quantity to be displayed and the *gridSize* as parameters, and outputs the positions for each SVG circle in a grid-like layout. The result is a dynamic grid of dots where the number of dots corresponds to the quantity value. One key advantage of this approach is its scalability, allowing the Tile to be extended to more complex representations (like 7x7 grids layouts).

For the remaining representation modes which don't fit into a grid layout - namely Rect, Disc, and Donut - custom algorithms were developed. These algorithms are designed for each to effectively translate quantity values into visually intuitive SVG shapes.

4.5 Speech API

For this implementation, we used the capabilities of Svelte's stores [12] and the browser's Web Speech API [8]. Utilizing stores enabled us to craft a stateful and user-friendly adaptation of the browser API. It simplifies the use of the API by allowing the user to configure only the relevant options: language, pitch, rate, and volume. To ensure persistence of user preferences, these settings are stored using the LocalStorage API[7] and loaded at the beginning of the application.

In addition, we also introduced a template system. This functionality allows the end-user to customize the spoken text by incorporating relevant variables. For instance, if a user

wishes to customize the dictation for the current level, say 'Level 9', they can write it as 'Level {lvl}'. The 'lvl' variable will then dynamically update according to the actual level, providing a more customizable user experience.

4.6 Fullscreen API

This implementation, like that of the Speech API, uses Svelte's stores and the native browser's Fullscreen API [6] to provide a simple toggle function.

We created a special API for this feature so we can ensure it works on as many browsers as possible. While the built-in functionality might vary between browsers, our API adapts to handle each case. This means you can turn on and off the fullscreen mode using a single button, regardless of the browser you're using.

4.7 Theme API

This implementation aims to be quite ambitious, thereby being more complex to put together. The main goal of the Theme API is to control the color scheme across the entire application, not just within the library's components. To accomplish this, we leveraged Svelte's powerful state management feature, known as stores, in combination with the CSSStyleSheet [4] and LocalStorage APIs.

The CSSStyleSheet API was employed to programmatically create a stylesheet that contains custom colors defined as CSS variables. This newly created stylesheet is then appended directly to the adoptedStyleSheets [5] array, which is part of the Shadow DOM API. By injecting our custom color variables into the *:root* of the CSS environment in this manner, we ensure that the entire application has access to these color variables without any complications.

To ensure the user's preferred settings are not lost, we employed the LocalStorage API. This way, user configurations are stored directly in the LocalStorage and loaded at the start of the application.

The implementation offers features ranging from providing predefined color schemes, generating new color schemes using just one color, to allowing customization of each color within the application. Developers can utilize the colors from this API directly in their applications, either by using the CSS variables generated by the library or the JavaScript variables accessible from the store.

4.8 Resources for Developers

A documentation [10] has been created that explains the use of each of the components and the APIs of the library. This documentation was developed with VitePress [14], and includes explanation of functionalities, documentation of interfaces, and usage examples.

In addition, a showcase type application [9] was built to show each of the components in action. The application is useful for both developers who implement the library in their applications, to see what the various components look like, as well as for those who develop the library, to have a space to visually see the results.

The documentation and the example documentation are automatically deployed to GitLab Pages when pushing to the repository in GitLab through a workflow.

Chapter 5

Validation

The validation of the library consisted of two essential stages. The first stage was direct validation with the users, the OATHs (Other Animals Than Humans) and their guardians, to ensure the functionality of the end product. The second stage was validation with developers to assess the ease of use, flexibility, and adaptability of the library in various application contexts.

5.1 User Validation

The validation of *InCA-CompareFast* was primarily conducted by Jérémy Barbay and the monk parakeets he lives with, Lorenzo and Tina. Additionally, valuable feedback was obtained from Claudia Ordóñez and the parrot Bonifacio. This validation phase was carried out using applications developed using the library's functionalities: *InCA-CompareFast*, a self implemented application, and other InCA-Lab applications' implementations, such as *InCA-ClickInOrder*.

5.1.1 InCA-CompareFast

InCA-CompareFast is essentially a reworked version of *InCA-WhatIsMore* using functionalities from *InCA-Utils*. In the *InCA-WhatIsMore* game, OATHs are required to compare different representations of quantities and identify the larger one. This game was reimagined with the newly developed library components in *InCA-CompareFast*, aiming to test the usability of these components and their effectiveness in an already established setting.

Rework Validation

This stage aims to verify the functioning of *InCA-CompareFast* as a rework of the *InCA-WhatIsMore* implementation. The goal is to see whether the application meets the projected

expectations and how the adaptation using *InCA-Utils* functionalities impacts the OATHs already accustomed to the *InCA-WhatIsMore* interface.

The validation is conducted by Jérémy, Lorenzo, and Tina. The initial expectation with the reworked version was to observe an improved adaptation from the OATHs to the new interface and to explore the potential of integrating automated difficulty levels. However, the results of the validation with *InCA-CompareFast* did not meet the expectations.

1. **OATHs Adaptation.** the OATHs showed difficulty in adjusting to the new interface, indicating that their familiarity with the original *InCA-WhatIsMore* interface may have hindered their ability to adapt to the changes in *InCA-CompareFast*. This observation suggests that significant interface changes in applications for OATHs should be introduced gradually or coupled with a more in-depth adaptation process.
2. **Automated Difficulty Levels.** The goal of introducing automated difficulty levels was not achieved with this application. This aspect indicates that automated difficulty adjustments present their own set of challenges and may require additional fine-tuning and testing to be effectively integrated into applications for OATHs.

These outcomes, while not in line with initial expectations, offer valuable insights. They indicate the challenges associated with significant interface changes and the integration of new features such as automated difficulty levels. This understanding will serve as a useful guide for future developments in creating applications for OATHs.

More Validations on InCA-CompareFast

Additionally, other validations were conducted that provided valuable insights into the development of *InCA-Utils*.

1. **Visual Indicator for Longpress Buttons:** The validations conducted with Jérémy, Lorenzo, and Tina revealed that there was no visual indication that the protected buttons needed to be held down to trigger them. With this feedback, a decision was made to implement an animation that filled a percentage of the button at the first contact. The current solution can be seen in the design section.
2. **Fullscreen Mode Does Not Work on iPhone:** The validations with Claudia and Bonifacio revealed that it is not possible to activate fullscreen mode on iPhones. The browsers' fullscreen API is currently not implemented on these devices, and there is no obvious way to make it work.

5.1.2 InCA-ClickInOrder

In the user validation phase of the InCA-ClickInOrder application, the OATHs were able to adapt well to the interface, easily engaging with the action buttons (exit and ready).

Additionally, a validation was performed by the guardian of the OATHs, which identified several design improvement points:

1. **Need for Completion Feedback.** There was a lack of feedback upon button press completion. This was particularly noted as being useful when the OATH's guardian's finger covered the entire button. While haptic feedback was already in place, it seems that it was either not sufficient or could go unnoticed on certain occasions. As a result, it was suggested to create an additional layer of feedback, this time auditory, to complement the haptic feedback.
2. **Option to Change Button Fill Direction.** There were instances where the guardian was unable to see the fill animation as its finger would cover the remaining unfilled portion of the button. This could be resolved by allowing the guardian to change the direction of the fill animation.
3. **Issues with Fullscreen Mode.** There were cases where the fullscreen mode did not operate as intended. For instance, when the application was left in fullscreen mode and the guardian was away, the screen was found locked upon return. Upon unlocking, the application was not in fullscreen, and an initial attempt to activate it did not work. However, a second activation attempt worked correctly. It is hypothesized that the fullscreen mode may lose its state upon screen lock.

As of the writing of this document, these issues have not been addressed in the application, and thus are left as "Future Work".

5.2 Developer Validation

For the developer validation phase, we utilized a survey tool, specifically Google Forms, to gather responses and feedback from developers using the library (creators of *InCA-Pop*, *InCA-ClickInOrder* and *InCA-ComBoard*). The purpose of this survey was to understand the difficulties they may have encountered while using the library and to gauge the usefulness of the documentation provided.

The survey consisted of the following questions:

1. What features of the library are you currently using?
2. How difficult was it to learn to use these features?
3. How obtrusive do you find the library in your development process?
4. Was the documentation useful to you?
5. Do you feel that the library has saved you development time?
6. (Optional) Suggestion of missing features or enhancements for the current functionalities.

The intention behind these questions was to gain a comprehensive understanding of the library's impact on developers' workflows, from the learning curve associated with its features to its potential time-saving benefits.

The results of the survey are the following;

1. All the developers are using the *TrainerButton* and *Fa* components. Some of the developers are also using the *fullscreen API* and the *ActionButton* component.
2. Developers didn't had problems to learn to use the features.
3. They also didn't think the library is obtrusive with its development process.
4. The answers got here are that they knew the existence of the library and it was helpful to implement the features.
5. The features in all cases improved the development time.
6. In this item I got 2 suggestion the improve the functionality of the library. The first is allow customization of the *TrainerButton* filling animation. A second review asked for a remodeling of the *Tile* component in order to handle directly the resulting SVGs (right now the SVG element is wrapped in a HTML *button* element).

Chapter 6

Conclusion

We conclude this report with a summary of our contributions (Section 6.1), a discussion of the potential issues with our work and conclusion (Section 6.2) and some perspective on directions for future work (Section 6.3).

6.1 Contribution

All the features proposed for the library were successfully implemented. The process of being involved in the development of *InCA-WhatIsMore* and the implementation of *InCA-CompareFast* was very important in identifying the essential functionalities these OATHs specialized applications should possess.

Also as a co-author of the publications "Measuring Discrimination Abilities of Monk Parakeets Between Discreet and Continuous Quantities Through a Digital Life Enrichment Application" [3] and "Can Monk Parakeets Compare Quantities Faster and/or Better than Humans? A Research Proposal," [2] significant insights were gained about developing applications specifically tailored for OATHs. This includes not just the technical aspects, but also ethical considerations in conducting experiments with non-human animals.

Despite the lack of examples of libraries implemented with Svelte, the project was structured without significant difficulties. The adopted workflows included deployment to *npm* and the documentation and examples' deployment to *GitLab Pages*.

At the beginning of this project, having a more extensive background in working with React, I was reluctant to use Svelte. However, after some exploration, I appreciated the simplicity and versatility of this framework. As a front-end developer, this discovery opened my mind to exploring other frameworks, such as Vue and Solid.js, extending my skillset and perspectives.

Overall, this project was a deeply educational experience that combined technical challenges with the need to respect and understand the non-human animals we work with. I believe that the *InCA-Utils* library can make a significant contribution to future research in

this exciting field.

6.2 Discussion

Originally, the focus of the thesis was on the implementation of *InCA-CompareFast* as a more complex version of *InCA-WhatIsMore* that incorporated automated levels. This initial approach failed, primarily due to the indifference of the OATHs (Lorenzo and Tina) towards playing with this rework, which complicated the testing and validation of the level changes. However, the functionalities developed in *InCA-CompareFast* were identified as valuable. They seemed to resolve many of the common problems encountered by InCA Labs applications. Consequently, the decision was made to pivot the research towards offering a set of tools specifically for user interface development, and this is how *InCA-Utils* was born.

For the reasons stated above, the first iteration of the library was released late in the semester when the developers of the other applications under development had almost completed their implementations. In most cases, they mainly implemented the *InCA-Utils* version of protected buttons (*TrainerButton*), the component with which they had the most difficulties. In some cases, other components, such as *ActionButton*s and the *Fa* component, were also implemented. The goal is for future InCA Labs developers to have access to the library from the beginning of their implementation efforts so that they can benefit from all the functionalities it offers.

6.3 Future Work

Several areas of future work have been identified during the development and validation of the InCA-Utils library:

1. **Backend Connection:** One of the most significant functionalities not yet implemented is creating an API to make it easier for applications to store data in *InCA-Backend*. Additionally, an option to save each user's personalization data in the backend should be developed. The aim is to enable synchronization of user data across various InCA Labs applications.

2. **TrainerButton Enhancements:** Two potential improvements for the accessibility of the protected buttons were identified during the validation phase. The first improvement is adding auditory feedback upon completion of the button fill. The second improvement is to provide an option to change the fill direction.

3. **Fullscreen API compatibilities:** At the time of writing this report, the fullscreen API is not supported by all browsers (e.g., Safari on the iPhone). Therefore, it is proposed to explore alternative solutions that prevent OATHs from interacting with buttons outside the game environment, such as the browser's navigation bar, status bar, or device navigation buttons.

Bibliography

[1] Rosalia Antunes-Foschini, Monica Alves, and Jayter Paula. "Blinding or Masking: which is more suitable for eye research?" In: *Arquivos brasileiros de oftalmologia* 82 (Aug. 2019). DOI: 10.5935/0004-2749.20190085.

[2] Jérémy Barbay and Fabián Jaña-Ubal. "Can Monk Parakeets Compare Quantities Faster and/or Better than Humans? A Research Proposal". In: *Proceedings of the Ninth International Conference on Animal-Computer Interaction*. ACI '22. Newcastle-upon-Tyne, United Kingdom: Association for Computing Machinery, 2023. ISBN: 9781450398305. DOI: 10.1145/3565995.3566027. https://doi.org/10.1145/3565995.3566027.

[3] Jérémy Barbay, Fabián Jaña-Ubal, and Cristóbal Sepulveda-Álvarez. "Measuring Discrimination Abilities of Monk Parakeets Between Discreet and Continuous Quantities Through a Digital Life Enrichment Application". In: *Proceedings of the Ninth International Conference on Animal-Computer Interaction*. ACI '22. Newcastle-upon-Tyne, United Kingdom: Association for Computing Machinery, 2023. ISBN: 9781450398305. DOI: 10.1145/3565995.3566040. https://doi.org/10.1145/3565995.3566040.

[4] Mozilla Developers. *CSSStyleSheet API*. https://developer.mozilla.org/en-US/docs/Web/API/CSSStyleSheet. Last accessed on 2023-07-23. 2023.

[5] Mozilla Developers. *Document: adoptedStyleSheets property*. https://developer.mozilla.org/en-US/docs/Web/API/Document/adoptedStyleSheets. Last accessed on 2023-07-23. 2023.

[6] Mozilla Developers. *Fullscreen API*. https://developer.mozilla.org/en-US/docs/Web/API/Fullscreen_API. Last accessed on 2023-07-23. 2023.

[7] Mozilla Developers. *LocalStorage API*. https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage. Last accessed on 2023-07-23. 2023.

[8] Mozilla Developers. *Web Speech API*. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API. Last accessed on 2023-07-23. 2023.

[9] Fabián Jaña. *InCA Utils Demo*. https://shockerqt.gitlab.io/inca-utils/demo/index. Last accessed on 2023-07-23. 2023.

[10] Fabián Jaña. *InCA Utils Docs*. https://shockerqt.gitlab.io/inca-utils/. Last accessed on 2023-07-23. 2023.

[11] Irene Pepperberg. "Animal language studies: What happened?" In: *Psychonomic Bulletin Review* 24 (July 2016). DOI: 10.3758/s13423-016-1101-y.

[12] Svelte. *svelte/store*. https://svelte.dev/docs/svelte-store. Last accessed on 2023-07-23. 2023.

[13] W3C. *Vibration API*. Accessed: 2023-07-23. 2022. https://www.w3.org/TR/vibration/.

[14] Evan You and VitePress Contributors. *VitePress*. https://vitepress.vuejs.org/. Last accessed on 2023-07-23. 2023.