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

INCA BAT:
BLIND ASSOCIATIVE TESTING WITH TOUCH INTERFACES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

PATRICIO JAVIER LÓPEZ TAULIS

PROFESOR GUÍA:
JÉRÉMY BARBAY

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
SANDRA DE LA FUENTE GONZÁLEZ

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE:
Ingeniero Civil en Computación
POR: Patricio Javier López Taulis
AÑO: 2022
PROFESOR GUÍA: Jeremy Barbay

INCA BAT: PRUEBAS ASOCIATIVAS CIEGAS CON INTERFACES TÁCTILES

Animal Computer Interaction (ACI) es un área de investigación que estudia interacciones de animales humanos y no humanos con tecnología. Entre otros tópicos, investigadores de ACI en habilidades sensoriales y cognitivas en no humanos, estudian las habilidades asociativas de estos, como entre una cantidad y un número escrito, una imagen y un dibujo, etc. Un problema fundamental en estos estudios es el riesgo de que el investigador humano, capaz de resolver las tareas solicitadas al sujeto de investigación, dé pistas (involuntariamente) sobre la solución correcta. Las pruebas a ciegas disminuyen aquel riesgo, pero de forma costosa: ¿Cuánto podría reducirse ese costo usando dispositivos digitales? Hemos formalizado, implementado y validado una aplicación con interfaz táctil que simplifica la configuración de pruebas a ciegas para experimentos con animales ya entrenados en el uso de pantallas táctiles. Para formalizar y validar la aplicación, contamos con la colaboración de una investigadora de ACI que diseñó un exitoso pero costoso sistema de pruebas ciegas no digitales, basado en un evaluador humano externo, tarjetas de papel, grabación de sesiones y posterior recolección de datos. La investigadora validó la usabilidad de la aplicación con cacatúas, y también los datos recolectados, declarando que puede ser usada para validar hipótesis sobre cognición animal aplicado en el estudio de habilidades asociativas. Además, ella declaró que la aplicación es ahora parte de la rutina diaria de entrenamiento de sus sujetos, dado que por sus cualidades enriquece la calidad de vida de estos.

ABSTRACT FOR MEMORIA
TO APPLY TO DEGREE OF:
Ingeniero Civil en Computación
BY: Patricio Javier López Taulis
YEAR: 2022
PROFESSOR: Jeremy Barbay

INCA BAT: BLIND ASSOCIATIVE TESTING WITH TOUCH INTERFACES

Animal Computer Interaction (ACI) is the research field that studies human and non-human interaction with technology. Among other topics, ACI researchers in non-human sensory and cognitive abilities investigate the associative abilities of non-humans, such as between a quantity and a written number, a picture and a drawing, etc. A core problem in such studies is the risk for the human experimenter, able to solve the task required from the subject, to (involuntarily) cue the subject about the correct answer. Blind testing setups diminish such risk but are costly: How much can such cost be reduced using digital devices? We formalized, implemented and validated a touch interface application that simplifies Blind Testing setups for experiments of small animals already trained in the use of touch screens. To help with both the formalization and the validation, we counted with the collaboration of an ACI researcher who designed a successful if costly non-digital Blind Testing setup, based on an external human evaluator, cardboard cards, session recording and slow data collection. The ACI researcher validated the usability of the application with cockatoos and the collected data, declaring that it can be used to validate animal cognition research hypotheses on associative abilities. In addition, the ACI researcher declared that the application is now part of the daily training routine of her subjects, due to its qualities for life enrichment.

Con cariño para mis padres Patricia y Roberto,
para mis hermanos y amigos, a mi querida Danielle
y a todos quienes me brindaron su apoyo para cumplir esta meta:
Son mi fuente de inspiración y perserverancia.
Sin ustedes no estaría hoy aquí.

Content

List of Tables

List of Figures

Chapter 1

Introduction

Animal-computer interaction (ACI) is a field of research for "*the design and use of technology with, for, and by animals*" [5]. It is a multidisciplinary field where engineering, design, computing, anthropology, behavior science, welfare science, etc. can work together with a common idea, connecting the benefits of technology with human and non-human animals. ACI has grown rapidly and consistently in the last decade, connecting researchers and practitioners from all over the world with different points of view on the same problem: understanding the possibilities and limitations of technology, for human and non-human animals.

Jennifer Cunha, an ACI researcher, studies advancing communication with birds [6]. Among other topics, she studies the possibility for birds to be able to learn how to read and write. For instance, showing a cockatoo two letters and pronouncing a word, the bird is able to choose which letter corresponds to the sound: that is an example of Binary Choice, a specific case of Associative Testing, that measures the subject's ability to relate two objects with the same meaning.

Cunha uses Android applications to train and entertain cockatoos. Mobile applications can be used for training them in the use of touch devices, to communicate and actively interact with her, and test diverse abilities of her birds. These digital games reduce the setup time and allows Cunha to try a wide a variety of applications to measure different abilities and improve their skills for using touch screens.

One of the most relevant abilities to study in animal cognition, is the associative ability. How much can they learn and what are the difficulties and challenges for them when they associate two different objects? To research about this, Cunha designed a Blind Associative Testing setup, using cardboard cards and recording each session, in order to watch it later to recollect data (Chapter 2).

It is necessary to improve this kind of research with non-human animals because human interaction can be involved in all the experiments, making its validation possibly biased by the researcher, who is the one that most of the time makes all the procedures. That is why we designed a Blind Associative Testing application that covered all the relevant points of Cunha's Blind Testing setup, to prove that ***the cost and risks of implementing a Blind Associative Testing, from setup to data recollection (create tests, play a BAT,***

save logs for future analysis) can be reduced using digital devices.

The designed web application (Chapter 3) emulates Cunha's Blind Testing setup, based on the same principles underlying her methodology: a human trainer (also called Teacher in this report) creates a set of cards and questions/asseverations related to those cards, chooses a set of one or more cards and one asseveration to ask to the non-human animal (also called Learner in this report). The Learner can choose one card after the question and then repeats the process.

We followed an agile methodology to manage this project (Chapter 4), to validate, fix and develop in a short time three important points ordered by importance: Blind Associative Testing, material creation and log creation. The methodology was based on a combination of SCRUM and Kanban, with sprints of 3 to 4 weeks to show and validate each point or a subset of features for each point. We started from the most important point, ensuring that the choosing card game is playable by the Learner and some of the most relevant settings to modify cards displaying and simple logging. The next point was adding more features, to create new tests with different cards and audios that are saved in client side, followed by improvement of logging system and data recollection. Once all important features were done, we finished improving user interfaces based on HCI principles to ease user interactions with the platform.

There was a informal validation on the end of each sprint, followed by requirements gathering for next sprints (Chapter 5). At the last part of the project, once the system was completed, we validated the three main important features mentioned before, software usability and desired future features for the software, with the help of Cunha. She declared her enthusiasm to use InCA BAT to validate animal cognition hypotheses on associative abilities, and she is now using the application as part of the core of her training sessions with birds.

Chapter 2

State of the Art

A core problem of research in animal cognition is understanding their ability to associate elements with their meaning. It has been studied more intensely since the 19th century, but having some credibility issues when researchers are directly involved in experimenting, like in associative testing, which concept we describe in Section 2.1. In order to solve that, Cunha (and others before her) designed a Blind Associative Testing setup, detailed in Section 2.2, that offers a possible solution to evaluate the associative abilities, of non-human animals, without bias. But even this good idea has practical limitations (its setup cost is too high), which can partially be alleviated by using technology. That is how with the help and experience of Cunha, in Section 2.3 we analyze the current applications and their problems when using it with cockatoos.

2.1 Animal Cognition and Associative Testing

Associative abilities are common in humans, associating sounds to things and symbols, or symbols to images. For example, learning a new language requires associating their graphemes with their sounds, and when we are learning the periodic table we associate the chemical elements with 1 or 2 letters. But in the case of non-human animals, there is not enough knowledge to judge the ability of each subject or each different species to associate different elements with same meaning.

Associative Testing helps us to measure the associative ability of individuals by displaying pictures, symbols, images or sounds and letting the subjects match them. For example, showing two cards with different symbols that have the same sound, let's say the sound for letter A and letter a, are the same, so we can associate them together because it has the same meaning. With this kind of test we can understand what are the limitations of each animal, or even species, in terms of association abilities. But, there is skepticism among the researchers of animal cognition about these experiments when a human tries to demonstrate that animals can understand, communicate efficiently, read, write, do the math, etcetera, because of the Clever Hans Effect [8]. In the early 1900, there was a horse called Hans that supposedly can solve arithmetical problems and other mental tasks by itself, but it was

demonstrated later that what he was doing was reading the human physical cues, not solving the mathematical problems.

Associative Testing does not train subjects to recognize certain element and associate it with others, but helps to determine which is the best way for each subject to associate elements, or if the subject can or cannot recognize any relationship between the elements. So, if the subject has the ability to associate elements, he can use this ability to learn new things by associating. Because of the *Clever Hans* effect, Associative Testing is not a reliable methodology by itself, requiring to add a Blind factor to improve the system, which we will describe in Section 2.2.

2.2 Blind Assosiative Testing (BAT)

A Blind Testing set-up minimizes human interactions to avoid the Clever Hans Effect [8], because the human is not directly interacting with the non-human animal during the experiment, and know neither the question nor its answer, so that the human lacks the information to decide which is the correct answer to the problem, and hence can not give any cue to the subject.

Cunha [6], and other researchers in ACI, have been interacting with cockatoos to realize experiments about communication with birds, using physical elements, like cardboard cards or touch interfaces. The experiment that we are focusing on, is what we call Associative Testing, where a human shows to a parrot cards with different symbols, the human makes a sound (a word or basic instruction in English) and the bird chooses one of the cards with its beak [1]. It may also be called Binary choice, because normally showing one card with each hand is the easiest way to make the test. The used cards have nothing in the back, so the human does not know which card is the correct one for the given question, and cannot give any (voluntary or involuntary) cues to the subject.

Blind Tests designed by Cunha are complicated for several reasons: They require help from other humans to make it easier, because it tries to isolate the test from the researcher; it is time consuming to get analyzable data from the experiments, and there is still a possibility that a bias can be given to the parrot without knowing it.

There are different kinds of non-human cognition tests. In this project, we focus on a limited type of Associative tests, where a non-human animal decides, among two or more choices, which one is associated with a given prompt. This experiment is not easy to set up and randomize without a computer, because of the possibility of bias in the randomization when it is done by a human. Currently, there is a lot of work behind each test, like recording sessions and watching it later to annotate results or similar settings, but there is still missed data and possible bias from the researchers on each experiment.

2.3 Animal Computer Interaction (ACI)

Cunha has trained her cockatoos to play learning games on smartphones and tablets, letting them play and receive rewards for correct answers, but using applications that are not designed for birds. Common problems are the use of many colors or animations that distract from the main task, characters that might be scary for a bird, strong sound or strong vibration.

The closest examples of associative testing games available for Android in Google Play are *Periodic Table Battle* [Figure2.1], *Math Mech: Basic Mathematics* [Figure2.2] or *Vocabulary Battle: Flashcards* [Figure2.3], but as we said before, these games have a lot of distractors for the main task, that a human can handle, but a bird might not be comfortable with. So it is needed to find a better approach to design an application for birds or small mammals.

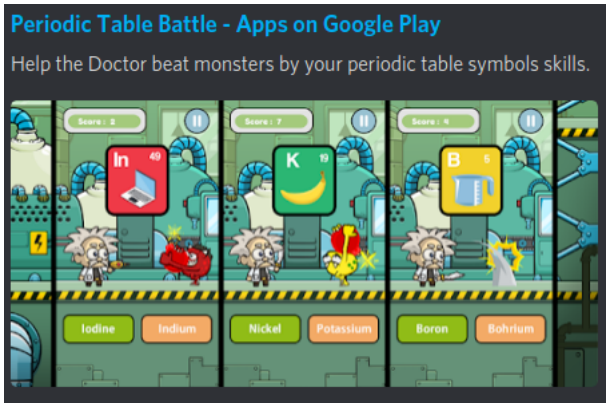


Figure 2.1: Periodic Table game for android in Google Play store, associates chemical element symbol (and a picture related to it) with 2 options. This game can be categorized as picture to text or symbol to text Associative Testing.

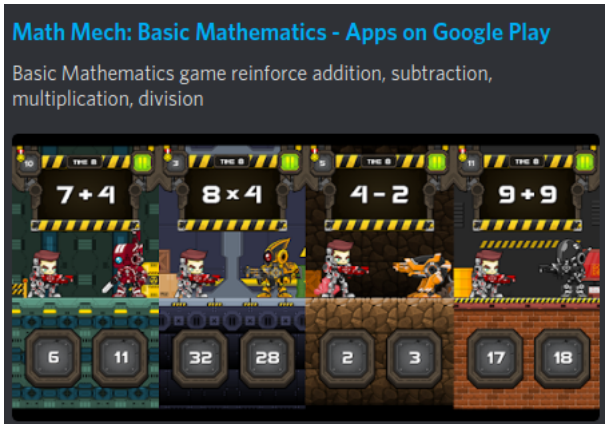


Figure 2.2: Math Mech game for android in Google Play store, associates an arithmetical problem with 2 options. This game is not directly associating a symbol, picture or a sound, but a meaning interpretation to another symbol.

Figure 2.3: Vocabulary Battle game for android in Google Play store, associates a picture of photo with 2 options. This kind of Associative Test is the closest one to our main purpose to achieve in this project.

Chapter 3

Design

The design of the application was inspired by the learner view of the application InCA Blind Comparative Testing (InCA BCT) [2] (illustrated by a screenshot in Figure 3.1), and the analogical Blind Associative Testing setup done by Cunha [6], using cardboard cards with words and letters of different colors and Foam Letters (illustrated by a screenshot in Figure 3.2). These designs are minimalist, specially InCA BCT with plain colors and trying to avoid most of the distractors that we are used to see in normal applications (like animations and colorful interfaces), and the reason is mainly because the users of such application might react with fear or might be distracted and not see clearly what is it supposed to do in the application. Because of that, we followed the same design philosophy in the prototype and final design for InCA BAT, with plain interfaces and minimal buttons, locking the access of the non-human user to specific sections in the application, the Learner View, and giving free access to the human user to configure the Learner view and add more cards and tests.

In order to understand the system and to design all needed interfaces, we describe a use case diagram in Section 3.1, that shows how the Teacher and Learner interact with the application. All these use cases generate or need Tests, Cards, and other objects, described by diagrams in Section 3.2, interacting with each other to setup a large amount of Blind Associative Tests (BAT). We continue describing the general requirements for the application in Section 3.3, but a critical part depends of the usability of the Learner views, where BAT are played, as a minimalist but complete cards selection game. We finish by a description of the options to the Teacher to create new material intuitively, adapt tests globally and retrieve all session logs in one click.

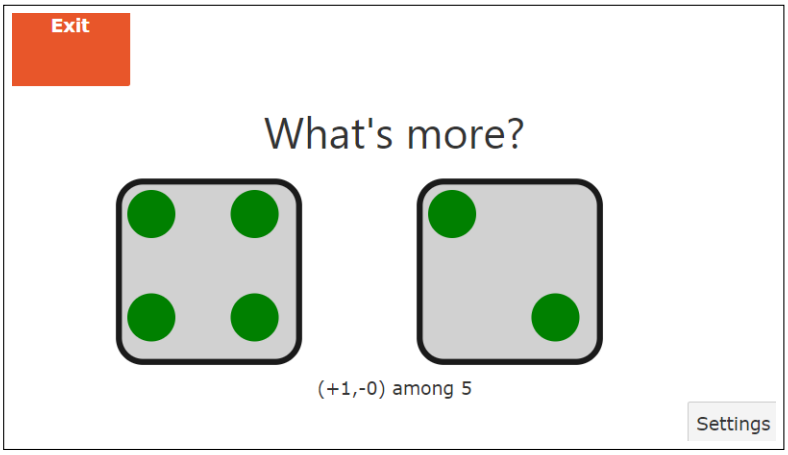


Figure 3.1: InCA BCT selection view: InCA BAT Learner view is based on the card position display of InCA BCT, showing a title on top of the view, an exit button, and cards ordered horizontally.

Figure 3.2: Foam letters used by Cunha on her researches about animal cognition, teaching how to read and write.

3.1 Use Cases

There are at least three major use cases for this application: start a Blind Associative Test (BAT) session, retrieve logs and create material. We separated each of those in more specific use cases, to clarify all the possible features to develop in the application. We also have two different users for this application, the Learner and the Teacher. The Learner is how we call the non-verbal animal (not restricted to non-humans), and the Teacher is the human researcher or caregiver of the Learner, both terms will be used in this project without differences. There are 7 use cases that the system must cover, described in Figure 3.3.

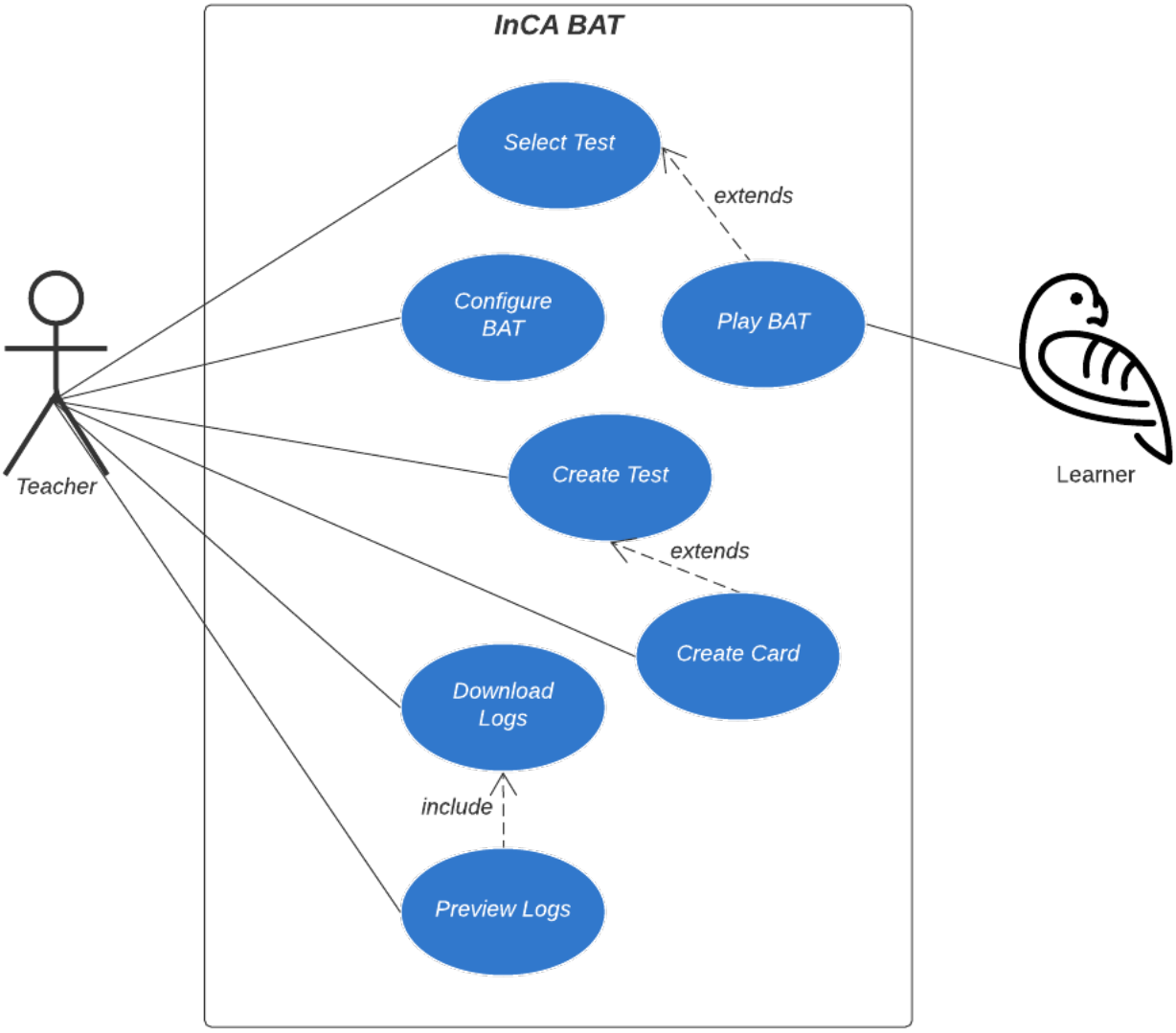


Figure 3.3: Use case diagram: Teachers (on left) can create cards and tests, configure the learner view and retrieve logs from BAT sessions. Learners (on right), symbolized by a parrot, can only access to play Blind Associative Test (BAT) sessions.

3.2 Objects Diagram

The application can be described by all the objects and their relations. It helps to clarify how Tests, Cards and Settings interact with each other, without adding actions. As seen in the use case diagram in Figure 3.3, a Teacher can create Tests and Cards, and Select a Test from the created Tests to allow a Learner to Play a BAT session. Figure 3.4, illustrates how a Test contains Cards, but there is a container object which represents that action, the Test Card, that also has an audio. This is needed to generalize a Test structure, with different behaviors of object associations (like different types of cards with different details, or image to image tests instead of image to sound test).

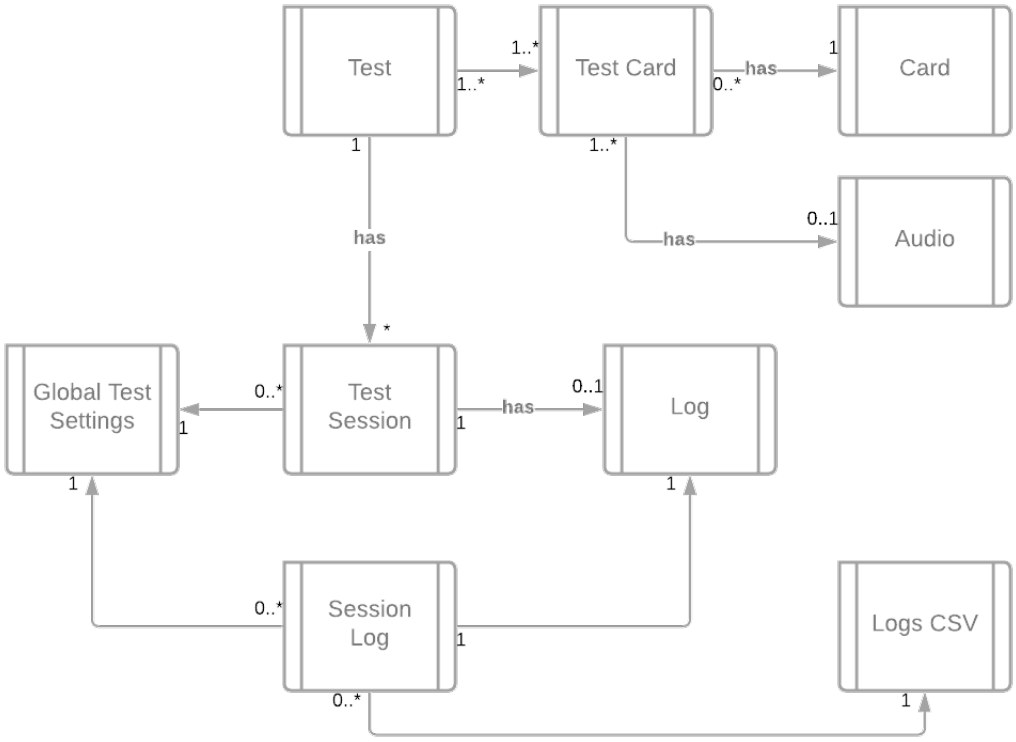


Figure 3.4: Objects diagram for InCA BAT: this diagram describes all needed objects, their relationship and interactions, without methods or functions.

3.3 Requirements

All the requirements were identified through interviews with the client, each of the interviews resulting in the addition of new requirements based on current version on that time, directly requested by the client or suggested by us. These requirements were categorized on important, normal and desired, where most of the desired ones were directly defined as future work because escaping from the objectives for this project in the given time available.

3.3.1 Blind Testing Basic Setup and Learner View

The Learner View is the most important part of the application, where the subject of study directly interacts with the touch interface, that is why we put special care on consistency of this view and all settings to modify the display of cards, sounds, colors and other details. *Blind Testing Basic Setup* refers to *Settings view* and *Test Selection view*, and *Learner view* refers to the part of the application that emulates a *Blind Associative Test* with cards and sounds.

There are three views that interact directly to cover a complete and highly configurable Blind Associative Testing: *Test Selection*, to choose and see details of each test among all the available tests for the user, *Settings*, to configure the game visual display and sounds for feedback and questions, and finally the *Learner view*, that includes all the logic of a Blind Associate Test (BAT), displaying randomly a configurable amount of cards on screen, allowing only one card to be clicked each time, giving feedback and repeat the process for 1 or more times (also configurable on settings view), after the session finishes it allows to restart the same test or go back to *Test Selection*. Specific requirements of this part are available in Annexed A.1.

Settings view includes the following configurable global variables that are applied to all the tests: amount of cards on screen, number of tests for each BAT session, cards separation, learner view background color, cards foreground and background color, feedback time delay, how much time hold the exit button while in BAT, show card text on screen, card height and width, enable/disable card sound message, and feedback sound.

Test Selection view allows the Teacher to select one of the tests, display test details and start new session with selected test. A *Test Preview* has 1 or more cards, and each card has 0 or 1 audio message that will be played on the Test Selection view if a card is touched, allowing to verify if the selected test is the desired or not. Finally there is a button to go back to menu or start the test, moving to the Learner view.

Learner view starts with a button on the center of the screen with the word “Start!”, once it is clicked the BAT session start. A BAT session has N tests, defined on Settings view as number of repetitions, that after T seconds, defined as feedback time delay on Settings, start a test. Each test in a session chooses randomly one card, from now the S+ card, from all the cards available on chosen Test on the Test Selection view, then choose randomly C-1 cards among left cards after removing the S+ card, where C is the minimum between the number of cards of chosen Test on the Test Selection View and the number of cards on screen defined

on Settings. After this process we have C cards, and one of them is the S+ card, and the test start playing the sound of the S+ card and showing on the screen all the C chosen cards. Once a card from the screen is picked, the next test start (including the delay time before showing the cards on the screen). When a session finished N tests, it concludes showing in the screen a summary of the correct and incorrect selections and 2 buttons: "Play Again?", to start a new session and "Go Back", to go to Test Selection view.

3.3.2 Logs Requirements

According to Cunha, data collection is crucial for research purposes, having a reliable system and enough data from each Blind Associative Test session. Without automatic log generation, generating such logs manually might take months. That makes this feature so relevant for the client and there are very specific requirements about the data, the format and the moment such a log is saved.

After each completed session, a complete log of session events is saved, and all the current settings setup for that test. The required session events to save in logs were required from the client, but adapted and improved based on the capabilities of the software: Session number, test number for current Test Session, Test name, correct choice (S+), Card audio message, Learner name, Caregiver name (CG), order of each card on screen from left to right, picked card name, picked card position, result, date, reaction time, background color, Card background color, Card foreground, feedback delay, card text (enabled or disabled, true or false), voice (true or false for enabled or disabled), feedback message chosen, Card height and width, Cards separation (percentage) and extra annotations. An example of required data is described in Figures 3.5, 3.6 and 3.7.

Logs can be accessed, retrieved and reset easily. They are saved locally and can be downloaded from the application in CSV format. To access session logs, there is a view that can be accessed from the main page (a visualization can be seen in Figure 4.15). Logs are displayed from last session to first session, with the last session on top. Each log has a table preview and a human readable summary, that can be shown and hidden with a button (each one).

	A	B	C	D	E	F	G
1	Session #	Test #	TestName	S+	CardAudioMsg	Learner	CG
2	3	1	touch the dot demo 2	2 dots card	touch the number 2	Ellie	Default
3	3	2	touch the dot demo 2	2 dots card	touch the number 2	Ellie	Default
4	3	3	touch the dot demo 2	dot card	touch the number 1	Ellie	Default
5	3	4	touch the dot demo 2	2 dots card	touch the number 2	Ellie	Default
6	3	5	touch the dot demo 2	2 dots card	touch the number 2	Ellie	Default

Figure 3.5: Logs table part 1: this session has 5 repetitions with 3 cards (white card, dot card, 2 dots card), 2 of them selectable (dot card and 2 dots card). TestName is the name of the selected test,Learner is Ellie and Caregiver/Teacher (GC) is set as Default, S+ represents the correct card and CardAudioMsg is the audio prompt.

	H	I	J	K	L	M	N	O	P	Q
1	C_0	C_1	C_2	C_3	C_4	PC_name	PC	R	Date	TimeR
2	2 dots card	white card	dot card			2 dots card	0	correct	18-12-2021 3:19	1738
3	white card	dot card	2 dots card			2 dots card	2	correct	18-12-2021 3:19	1505
4	dot card	white card	2 dots card			dot card	0	correct	18-12-2021 3:19	1565
5	dot card	2 dots card	white card			2 dots card	1	correct	18-12-2021 3:19	1363
6	dot card	2 dots card	white card			2 dots card	1	correct	18-12-2021 3:19	1242

Figure 3.6: Logs table part 2: this session has 5 repetitions with 3 cards (white card, dot card, 2 dots card), 2 of them selectable (dot card and 2 dots card). From `C_0` to `C_4` the names of the cards ordered from left to right, picked card (PC) represents the index of selected card (0 to 4) and its name in PC_name. Column R is for result of picked card, Data is for exact time when touched the card and TimeR for time reaction in milliseconds.

	R	S	T	U	V	W	X	Y	Z	AA	AB
1	BgColor	CardBg	CardFg	FbDelay	CardText	Voice	FbMsg	CardH	CardW	CardSeparation	Annotations
2	white	white	black	1500	false	true	yes/no	180	160	10%	
3	white	white	black	1500	false	true	yes/no	180	160	10%	
4	white	white	black	1500	false	true	yes/no	180	160	10%	
5	white	white	black	1500	false	true	yes/no	180	160	10%	
6	white	white	black	1500	false	true	yes/no	180	160	10%	

Figure 3.7: Logs table part 3: all settings that modifies BAT sessions are described in last 11 columns of logs, these are from left to right background color, card background color, card foreground, feedback delay (in milliseconds), card text (enabled is true, and disabled is false), voice (same as card text), feedback message, card height (in pixels), card width (in pixels), cards separation and annotations. Annotations are always empty.

3.3.3 Material Creation

Due to the large amount of possible tests to create, to study different topics of image-sound association, we designed the first version of this application with two simple tests based on touching a dot and decided to give freedom to create new tests, with combinations of custom cards and sounds.

A Card is an image that represent a choice for a Blind Test, and there are three possible Cards types: Image card, that can be set with the image url, dice Cards (preset SVG images with dice shapes), and Text cards, that can be adjusted adding a text and adjusting it to the card size. Each Card can be created and deleted directly from the application (except for diceCard), in a menu that can be accessed from the main page (see the button *My Cards* on Figure 3.10).

A Test is a collection of Cards, with 0 or 1 Audio associated to each card. There is only one type of test, Image-Audio Association Test, and these can be created from the view to access local Tests (see button *My Tests* on Figure 3.10). The needed steps to create a Test are listed below:

1. Select 1 more cards, if the needed card is not available, it can be created with a button

available in the same view.

2. Add a short text description for each Card, this description will be transformed from text to speech and linked to the Card in current Test. If no text description is added, there is no audio linked to the Card.

3. Add a name for the Test, this will be displayed on Test Selector view to find a Test.

4. Confirm creation. In this step, if confirm is pressed, all cards with description will link an audio with the text as a content of the audio and set as *Selectable*, if there is no text description, there is no audio linked and is set as *non Selectable*.

Selectable cards in a Test can be chosen as correct choice in random card selection. This behavior allows to set *filling* Cards for some tests, it means that the card image will be shown but never will be an option to click. This setup simplifies Test creation and gives an useful tool for *ignoring* some cards from the card selection.

Tests and Cards can be listed and deleted from their main views, protecting the system for cascade delete, it means that a Card cannot be deleted if there is a Test using it, and if a Test is deleted, it does not delete used Cards. This behavior is just available for Local Tests and Cards (created in the device).

3.3.4 Other requirements

Cunha requested and remarked the importance of mentioning her Parrot Kindergarten project in the application, adding a hyperlink to it in the main page of InCA BAT that is visible and redirect to her web page located in `https://parrotkindergarten.com/`. We also proposed to add a view for this purpose in a section named *About*, that can be accessed from main page and it also includes a mention to Parrot Kindergarten and a brief summary of the project to clarify intentions. A description of this view can be seen in Annexed B.2.

3.4 Learner Interface

The Learner interface covers the main requirements for Learner View seen on Section 3.3.1, allowing to start a Blind Associative Test Session, with all the settings from the Settings view (details on Section 3.3.1). This view was designed as a minimalist but dynamically extensible interface, allowing the user to modify as needed in order to create very specific situations to Test cognitive abilities of non-human animals.

To ensure that the Learner has an extremely limited interaction with the interface, we designed the flow of the Learner View (Figure 3.8) with no more than two actions while in game: Pick a Card and Replay. This setup not only limit the Learner but allows it to play by itself without Teacher interaction with the application. An example of a test session view can be seen in Figure 3.9.

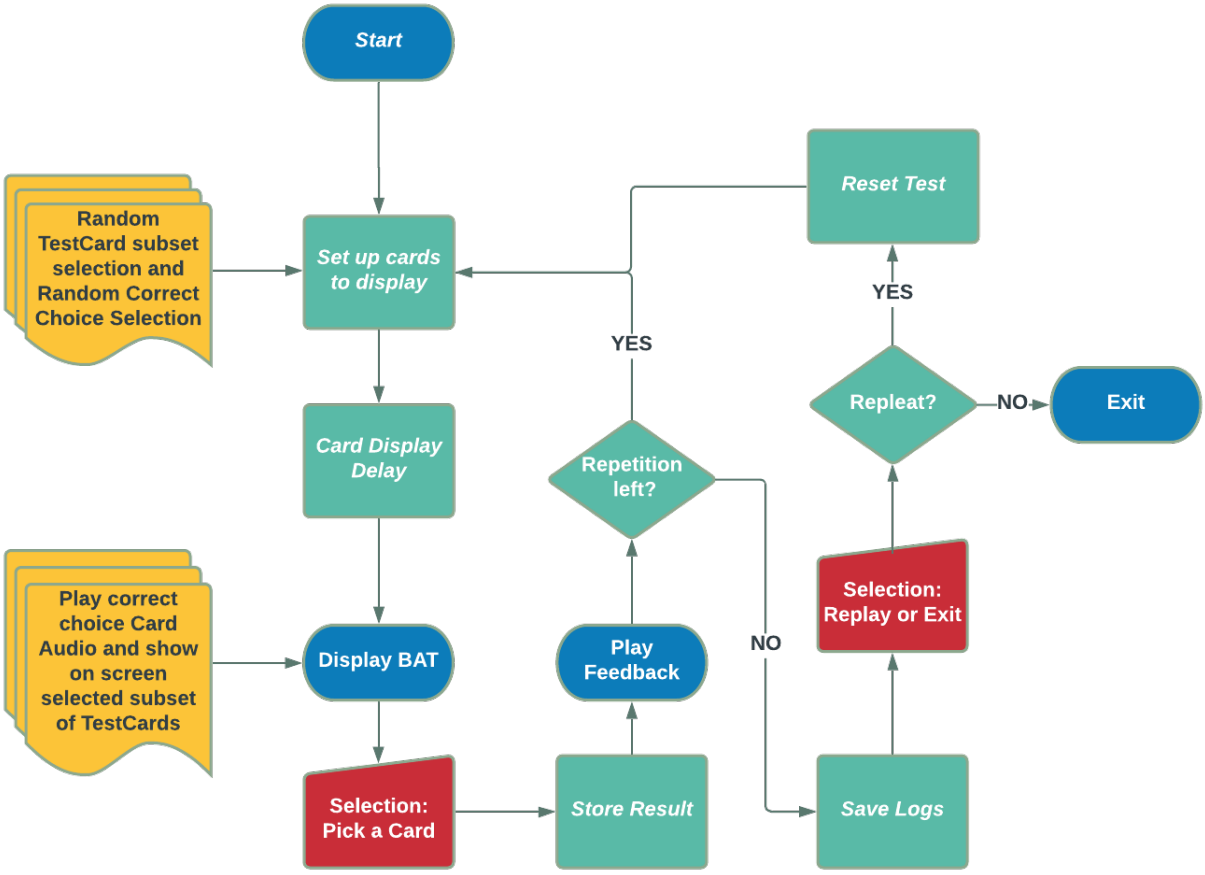


Figure 3.8: Play BAT Flowchart: This flowchart describes the Learner interaction with the Learner interface. Comments in yellow, actions in blue, user actions in red, internal process in green.

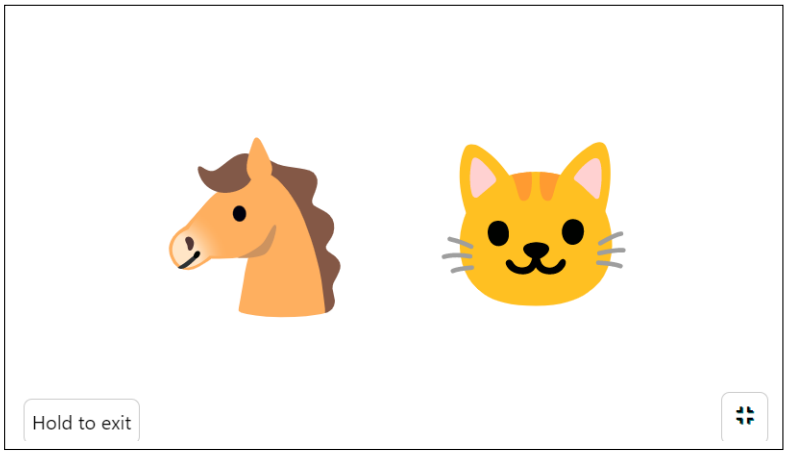


Figure 3.9: Play BAT Test Session example view: This example shows 2 cards on the screen, with default settings (50% card separation with white background, foreground and card background). To ensure Learner correct interaction without distraction, there are limited elements on screen, such as cards and two small buttons on bottom corner.

3.5 Teacher Interfaces

When we refers to Teacher Interfaces, we mean all the views that can help a Teacher to setup a Blind Associative Test, including: Main navigation page, information about the project, local logs, card management, test management, application settings, BAT settings and test selection. We designed these views thinking of a Teacher, giving the possibility to create new materials and navigate in few taps from one view to another, without hidden views and making it clear and smooth. All the views can be accessed from the main page as seen in Figure 3.10. In Annexed F.1 we can see a detailed flowchart of the the application navigation.

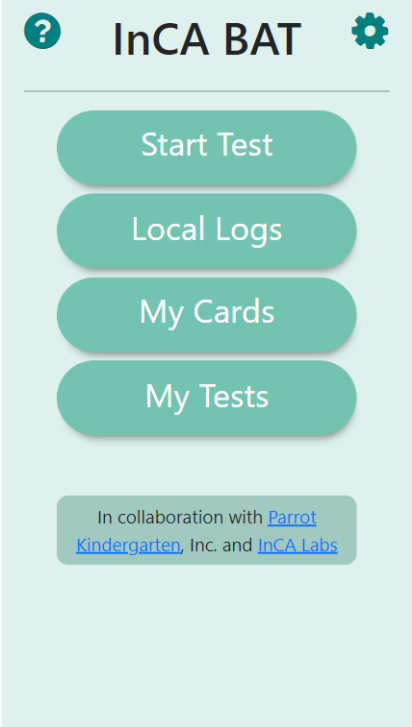


Figure 3.10: InCA BAT Main page: There are 6 buttons to access all the sections of the application, on top left the access for About view, top right Settings, followed by test selection, logs view, local cards management, and local test management. On the bottom the collaboration declaration.

3.6 Data Model

InCA BAT data model includes all the requirements described in Section 3.3, included Logs, Settings, Test Sessions, Test and Cards. We describe in the following entity-relationship model (Figure 3.11), a data oriented model based on the object diagram seen in Figure 3.4, assuming a TestSession as a container for all the Logging information needed by the client.

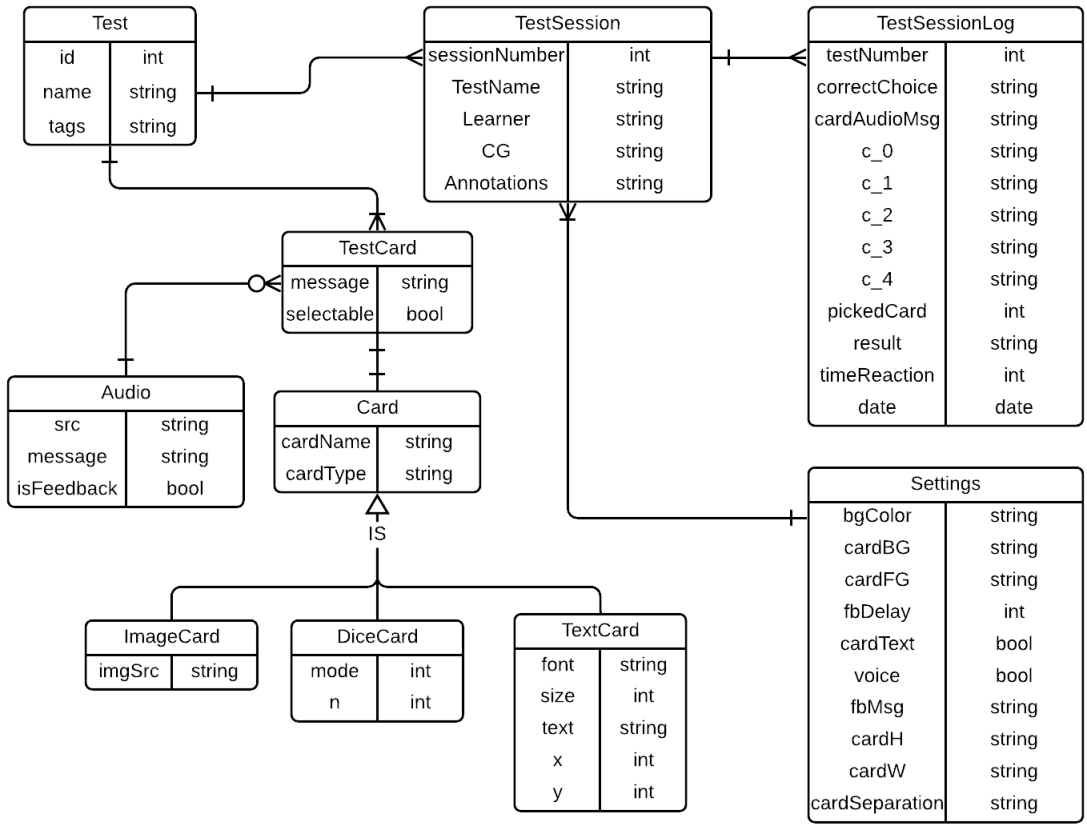


Figure 3.11: Data model: Containing a total of 10 entities, this diagram manage Tests and Cards (which can be 3 different types of cards) and Logs (with TestSession, TestSessionLog and Settings).

As we can see in Figure 3.11, each Test has one or more TestCards. TestCard is the entity that contains the logic for a Blind Associative Test among an image and an audio, where a Card extends to DiceCard, ImageCard or TextCard, each one including different creation details. This design allows the creation of a large amount of Cards, but not limited to it, because it allows to extend the model to add more card types that might not be considered in this project. This model (Figure 3.11), also allows storing session logs in a database, saving all the requested data (see Section 3.3.2) from each session to further analysis.

This model does not include users, because it was designed to interact with a frontend application that does not have login options. But it is a simple model that can be easily extended to include users, adding a new entity User or Teacher that is related with Test, Card, Audio, and/or TestSession.

Chapter 4

Implementation

The implementation of the InCA BAT software followed an agile methodology of work, detailed in Section 4.1, to have small versions of the application for early validation. That is why we chose technologies like Svelte, described in Section 4.2, to develop and deploy fast the *frontend*, Section 4.3, and *backend*, Section 4.4. We explain in detail how to store variables on the client side, Section 4.3.2, and how to use it in Learner and Teacher views, with an extensive description of these interfaces development, Section 4.3.3 and Section 4.3.4 respectively. We continue describing how we partially implemented an API, to interact with the *frontend* in Section 4.4, its importance to storing data, and generating audio files for the text to speech functionality that is a core functionality of the application, Section 4.4. We finish the implementation description with our deployment methodology in Section 4.5.

4.1 Work Methodology

We followed an agile methodology in this project, using Kanban and Scrum principles, specifically using 5 lists of tasks:simple urgent, complex urgent, simple non-urgent, desired but complex and things that are done. Each completed task was moved to the *done* list. Every 2 to 4 weeks the list was updated, presenting to the client the *done* tasks, to have an informal validation in a period of 2 weeks, and tasks still on the list were moved, and new tasks were added, adjusting priorities. This methodology was implemented to develop as many features as we can in the shortest possible period, and moving features that were underestimated replacing it by simpler ones while looking for other solutions. This allowed us to develop and validate in parallel, and make fast adaptations whenever is needed. We delivered a minimal version in week 6 and small new features every 2 to 3 weeks, with personal meetings with informal interviews to gather requirements, and a final validation in the last week.

We developed InCA BAT in two parts, using software versioning with Git and Github repositories. These two part are a *frontend* repository for main InCA BAT client side oriented application, and a *backend* repository for an API that communicates to a database and a text to speech integration, to create audio files for Tests created in *frontend*. Using versioning allowed us to have control of breaking changes and working organization, separating

functionalities and dependencies that makes the software easier to extend for further developers. The main objective was having an stable *frontend* with no dependencies of a *backend*, and the least part of the time was dedicated to implement an API structure, extensible but simple, that can adapt to the *frontend* application.

Due to constant discovery and change or priorities, the designed model for database was not strictly implemented, but a subset of properties that are adaptable to create a more stable version.

4.2 Technologies

We chose the following technologies to develop InCA BAT, based on the objectives, current knowledge, learning curve and flexibility for breaking changes of requirements and/or objectives:

1. Svelte: for *frontend* development.
2. Flask: for *backend* development, oriented to simple and fast API creation, using python 3.
3. MongoDB: non-relational database to store data with backend API.

We decided to create a web application that can be tested easily in many devices with different operative systems, to avoid possible installation issues, and to ease user interaction with new versions of the application (due to the possibility of a large a amount of version changes). Svelte allows us to create a very flexible, simple and reactive software, that will run in any device that has a browser. Svelte can be structured with components like other frontend popular libraries, allowing to extend the system, and reuse code reducing time of development that can be used to test features with low time cost. The only negative thing is that we are restricted to browser capabilities, but we considered it not important for this project, because InCA BAT design principle is being simple and non-human animal oriented.

Svelte is an easy to learn framework that allowed us to develop and try fast, something needed because of the low knowledge we had about the topic (animal cognition) and the possible reactions of birds using the application.

We decided to build a small backend API using Flask, due to low configuration needed and fast deployment. The other considered option was Django, also using python 3, but this option required more development time and has more flexibility restrictions, taking more time to modify an endpoint compared with Flask. Even with this setup, this project did not allowed us to work enough in the backend system, due to large amount of *frontend* requirements.

We decided to use a mongodb, a non-relational database instead of a relational database, because of the constant design changes in the middle of the project while adapting to user requirements. We considered risky to have a relational database, because it is not easy to

modify the model structure many times during the project. The benefits of mongodb are the flexibility to save information in JSON format, that is closer to the *frontend* design with javascript based language, it makes easier to submit and retrieve data to an API, making it even faster to develop. Disadvantages of using non-relational database are not relevant in our situation, but might be an issue for future development if querys gets more complex.

All these technologies makes InCA BAT source code highly flexible to further modifications once the structure of the system becomes mature, and data structures gets more stable.

4.3 Frontend

The main objective of InCA BAT is having an useful system to setup Blind Associative Tests on digital devices. It means that the frontend of this application had to be reliable and complete, allowing the user to create a large amount of different tests. Because of this, we focused the design and development on client side features, validating a large a amount of test cases using client side storage (Section 4.3.2), included logs saving and setting options. We ensured that Learner view is consistent, showing correctly each card from each created Test (Section 4.3.4), randomizing Test choices without human interaction and keeping simple this interface (Section 4.3.3). Source code is available in `https://github.com/plt1994/inca-bat`

4.3.1 Source Code Structure

InCA BAT *frontend* code structure is component oriented, as it can be seen in Figure 4.1 in `src/components`, Svelte code is encapsulated in `components` directory, separated in CardComponents, Games, LogComponents, SettingComponents and Utils (see complete detail of files in Annexed C). This structure makes the code extensible, for further developments, for example to add image to image associative tests.

4.3.2 Local Storage

InCA BAT stores a large number of global variables, used among all the application, such as Settings parameters and navigation pointers. These variables are easily managed thanks to Svelte Stores system, that allows to use and update variables from any part of the application. But it resets everytime the web page is reloaded, making it a partial solution to store data, because it is not persistent among sessions. To solve this problem, we used svelte Stores with local storage from browsers, allowing to save data and keep it from one session to another from all over the application. An example for storing a `username` variable persistent among sessions can be seen in Figure 4.2, and its usage in Figure 4.3.

We use svelte stores to store all variables in InCA BAT, such as local Settings, Logs, Tests and Cards. The only things that are not stored locally are images and sounds, that

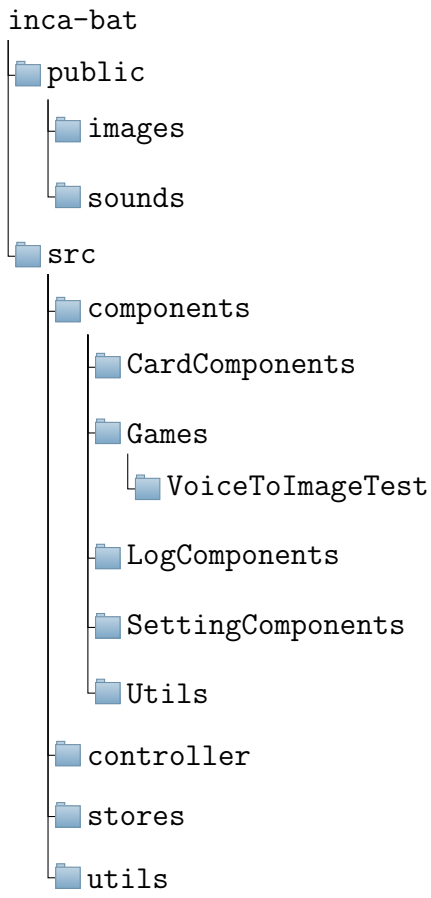


Figure 4.1: Code structure directory view: InCA BAT frontend project code structure has components that manage the views, controller that manage data model abstraction, stores for global variables and utils for any other utilities.

are directly obtained from the source of the url. Logs, Cards and Tests variables structures are `JSON` like strings, parsed to `JSON` when read from the localStorage, allowing the system to manage local data without the need of persistent queries to an API.

4.3.3 Learner View

Learner view has 4 states: *pre-start*, *waiting*, *running* and *post-game*. Each one described in Figure 4.4, Figure 4.5, Figure 4.6 and Figure 4.7 respectively. *Waiting* state happens just before each *running* state and establish a time delay, from 0 to 3.5 seconds, drafting a subset of cards to display on *running* state (flowchart diagram details in Section 3.5). When the Learner touch a card from the screen, a test log is generated and add a counter to number of repetitions. If number of repetitions is equal to Settings number of repetitions, *post-game* state start, saving local logs and asking if play again or leave the game.

A problem we discovered from prototyping and first validation for Learner views with real birds, was that it is easy for the Learner to accidentally hide the application once the test started by simply scrolling down, and even in some devices that action refreshed the web

```
1  import { writable } from 'svelte/store';
2  ...
3  const storedUsername = localStorage.username
4  ...
5  export const username = writable(storedUsername || "Default")
6
7  username.subscribe((value) => {
8      localStorage.username = value
9  })
```

Figure 4.2: Save and persist username among sessions: in stores.js file, we imported `writable` from `'svelte/store'` to use Svelte stores, and we retrieve the last value of username saved in last session in line 3. In line 5 we export the username value to use in other files. In lines 7 to 9 we update the value of local storage, everytime variable `username` changes

```
1  <script>
2  ...
3  import { username } from 'stores/stores.js';
4  ...
5  </script>
6  ...
7  <div>
8      <input bind:value={$username} />
9  </div>
10 ...
```

Figure 4.3: Usage of store variable, example for username in Settings.svelte: we imported `username` from stores and bind it in a input html tag. This will dynamically change username value from stores, each time the username is changed in the input.

page and the application went back to main page. To solve this, we ensured that in every screen, once we enter to Learner view, we automatically set full screen, with fixed screen width to avoid any possible scrolling to any side of the screen, producing a sort of *screenlock*.

As we can see in Figure 4.6, there are 2 small buttons on bottom left and bottom right corners. Bottom left button allows the Teacher to stop a test before it concludes, holding it for few seconds (configurable on Settings view). Bottom right button allow us to enter and leave full screen mode if clicked. When leaving full screen mode, the current test is paused showing a *pause* view (Figure 4.8), with one button to continue the test. When button for continuing the test is pressed, a new random draft is automatically requested (that might be different from the one before the pause), all the other elements do not change.

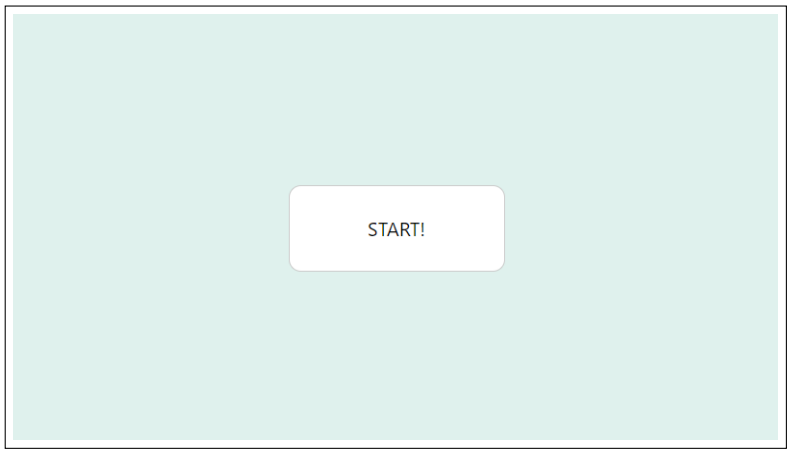


Figure 4.4: Learner View Pre-Start: this is the starting screen when entering to a new BAT session. From this point is designed to be used by a Learner without support.

Figure 4.5: Learner View Waiting: this view shows the background color only (white in this example) and no cards, for 1.5 seconds that is modifiable on Settings. In this point draft a subset of cards for next view.

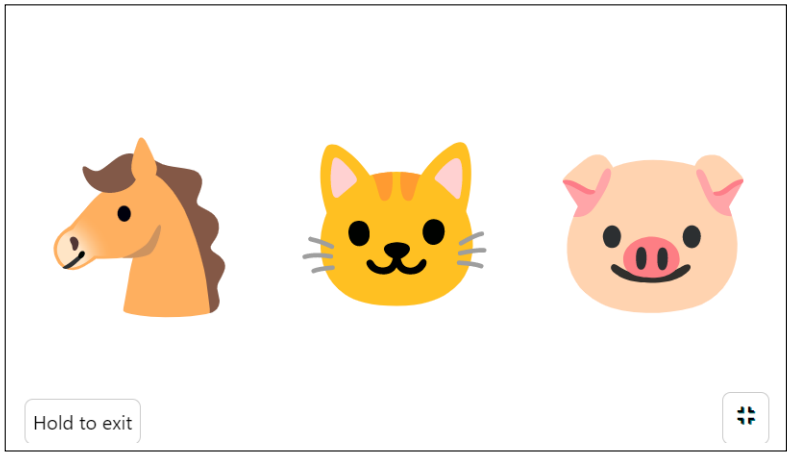


Figure 4.6: Learner View Running: this view shows drafted cards and prompt audio from one of the cards to be selected. Once one of the cards is selected it goes to waiting view if there are repetitions left. Hold to exit button and Exit full screen button on right.

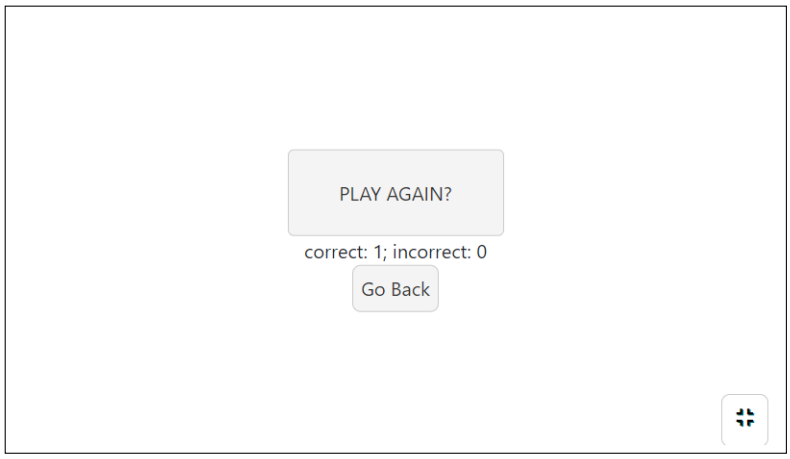


Figure 4.7: Learner View Post-Game: When no repetitions left for current current session we can see 3 buttons on screen and a brief summary of the test results.

Figure 4.8: Learner View Pause: If bottom right button is pressed, we leave fullscreen view and pause the game. If continue button is pressed, goes directly to running view with a new drafted subset of cards and audio prompt.

4.3.4 Teacher Views

The teacher views are all the views to configure cards, test, application seetings and the learner view (to setup global settings for all blind associative tests), and retrieve logs. In this section we show all the implemented views and details about each one, following design described in Section 3.5.

Navigation

To move from one view to another, we implemented a system of Links, views Stack and views internal name. Each section (settings, card display, card creator, test display, test creator, logs, test selection and learner view) has a key name, and there is a variable for current section that is updated each time we move from one view to another using Links. Each time we move to another view, last visited view is added to a Stack, allowing us to go back from one point to another, but not to go forward. This feature allows us to avoid routes changes, making the application more *static* and purely *javascript* dependent.

To navigate from one section to another we added a footer on every teacher view , letting us go from anywhere to main menu, but changing content and number of buttons in some views depending on the interface needed actions. For example, on *My Tests* and *My Cards* sections we included a *Back* button to keep a constant for Test and Card creation, that requires 2 or more steps to create a new Test or Card, but in Settings, Test Selection and Logs view, we do not need a back button because it goes directly to main menu. To keep a non-high mental model breaking, we put all the buttons with same shape, home button has an icon of a house, and back button is always on the left. Other actions are always added to the right side of the footer. A list of footers that can be seen in InCA BAT are described in Annexed B.1.

Settings

Settings view implementation used defined variables in stores, to globally apply it, displaying them and adding options to modify them. All needed variables directly modifiable in settings were obtained from Section 3.3. Settings view is composed by 2 components, *Settings*, for structure, and *SettingsOption* for each option display (components structure in Annexed C). Settings also have a footer with a *Preview* button, where it is possible to preview the Learner view with currently selected Test and current setup on Settings, just ignoring the number of repetitions (there is only 1 repetition in Learner Preview mode). Logs can be cleaned in this view with the button *Clean Logs*, but needs to hold the button for 3 seconds. These details can be observed in Figure 4.9.

BAT sessions (learner view), apply all these settings in real time and keep from one session to another if the web page is refreshed, but these settings are just available on the user browser and device it was setup, because we did not implement users that can migrate data to one device to another, or any other mechanism. If users needs to apply same settings

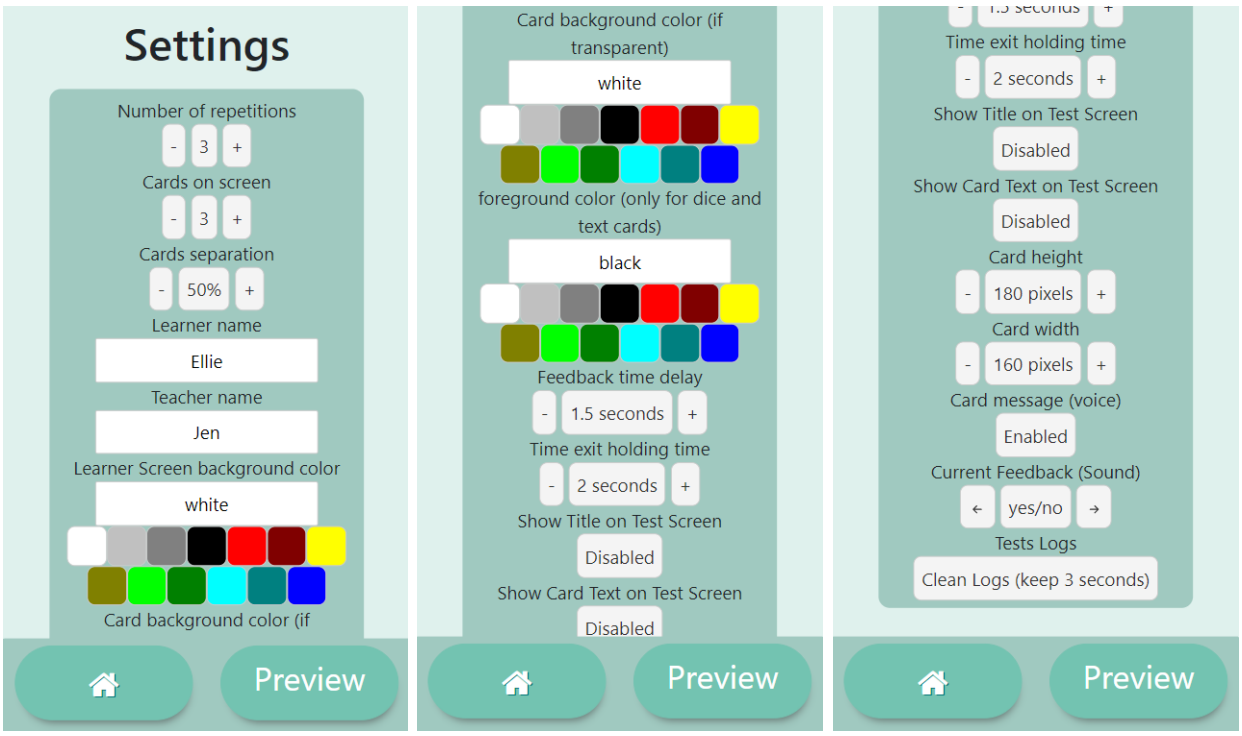


Figure 4.9: Settings view: from left to right the Settings view are scrolling down. First option is Number of repetitions and last option is Tests Logs cleaning.

in many devices or even many browser in one device, it needs to be done manually.

My Cards

To display list of local Cards, create new ones and delete them, we created the section *My Cards* accessed from main page. List of cards with preview and delete option are in the same view, we call it *display view*, see Figure 4.10. One or more cards can be selected to delete, showing a confirmation message with the cards that will be deleted, and restrict card deletion if the card is being used in one or more Tests, more details in Annexed B.3.

If the user wants to create a new card, the *Card Creation View* can be accessed from the *New card* button on cards *display view*. There are two options to create cards (Figure 4.11), Emoji Card and Text Card. Emoji card creates cards adding an image url or choosing an emoticon from the list, see Figure 4.12. Text Cards can be created using a text, and allows the user to set the size, font and position of the text in the card, detailed Text Card creation views are available in Annexed B.4. After setting up all the needed information to create a card, there will be a small button under the card to save it, and other card can be created if wanted.

Each time a card is created or deleted, the global variable that stores cards in localStorage is updated from a `controller`, and each time a card is displayed, the same `controller` search it from the localStorage, using the stores. This system isolate the features for obtaining and updating data, allowing improvements for new developments in this software.

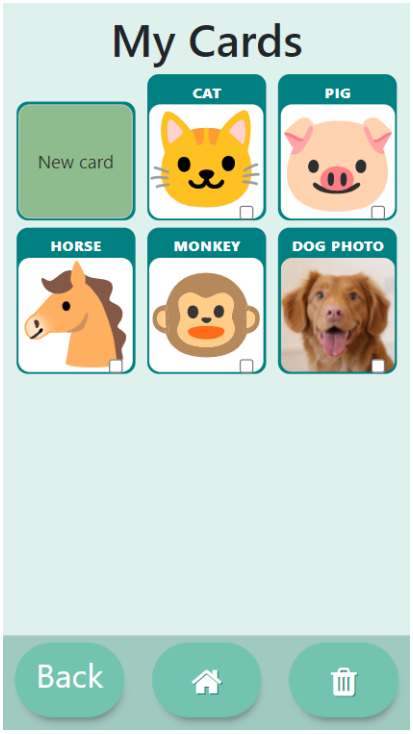


Figure 4.10: Card list view: Card display view, on top left the button to access card creator. On bottom the footer to go back, to main menu, and delete selected cards.

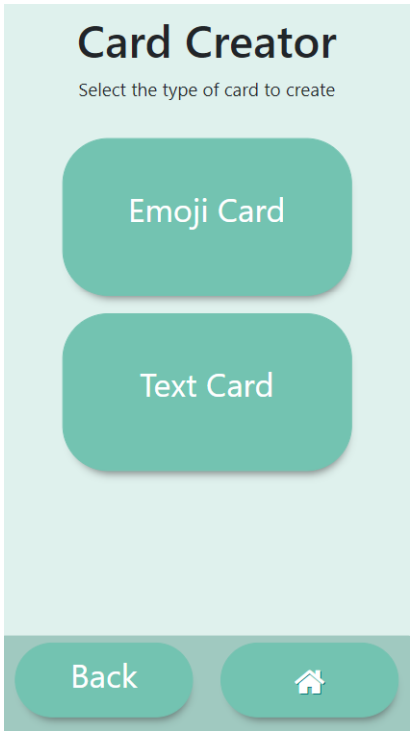


Figure 4.11: Card Creator Selector: this is the first view when selecting to add a new card. There are 2 options, text cards and emoji cards. Emoji cards also works for image cards.

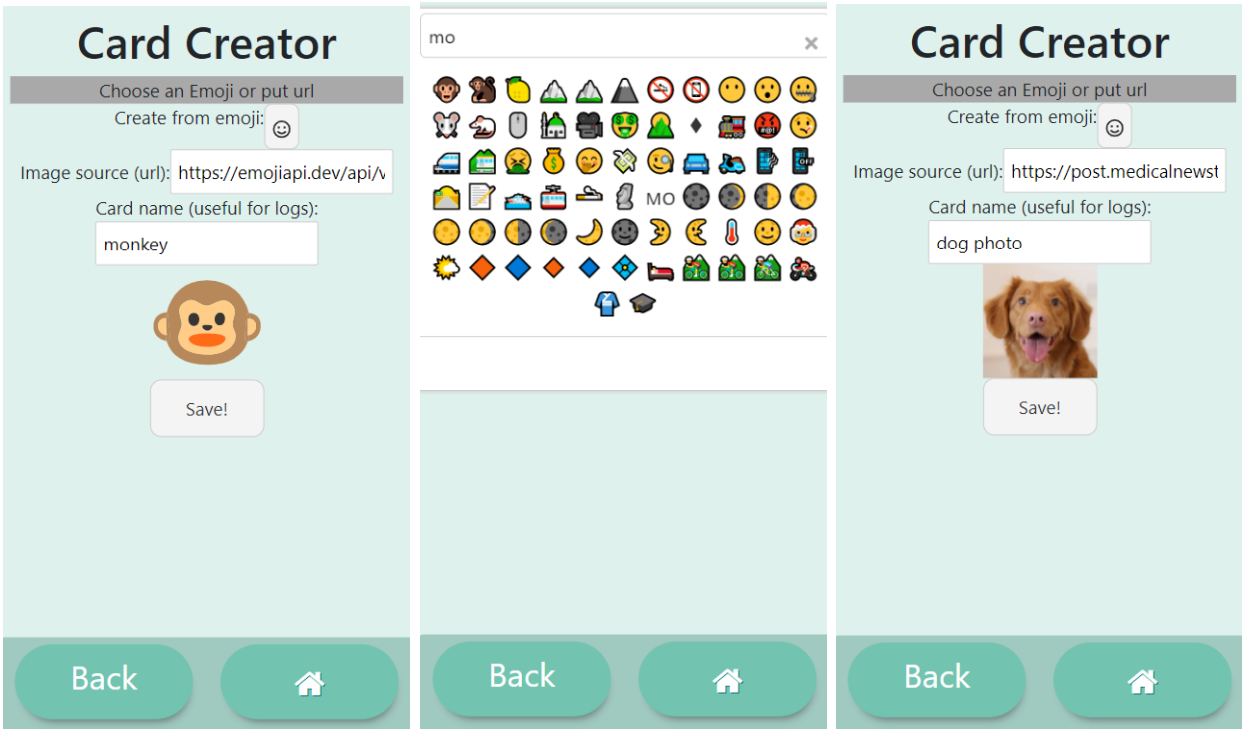


Figure 4.12: Card Creator: Example for emoji card creator selecting a monkey face emoji on left. In the center the emoji selector with emoji searcher. On right an example for image card creator adding a source image.

My Tests

Similar to Card *display view* and delete options, we implemented a view to display current local tests, with delete and create options. The test display can be accessed from main page selecting *My Tests*, and it will show on the screen all the created tests in local client browser, see Figure 4.13. Each test preview shows its name and all the cards added to it, and if a card is tapped it will play its sound if available. Each test can be deleted when the small trash can is selected, with a previous confirmation, details in Annexed B.5. We decided to add two buttons for create tests, because its a main feature that needs to be easily accessed, and first thing an user will see is top center of the screen, but bottom right is the natural position for users that use a mobile device.

Test creation requires some more steps compared with card creation, and interaction with backend API to obtain a voice message for each card, and it is going to be explained in detail and how did we implement it. Test Creation starts when tapping *Create* button in top of the screen or in footer, taking us to *test creation view*, see Figure 4.14, where all available cards (local and global) are displayed on screen, but also can create a new card if needed, taking us to card creation view. In the first step we choose one or more cards, and click on *Next* button, then we set a message to each card, this message will be transformed from text to speech and it will be played on Learner view as a question to make the association between the card and the sound. If a card does not have a text message, that card will be non selectable on card random drafting for BAT session (Learner view). Next step requires to setup a name for the test and finish with creation confirmation, in this part there is an

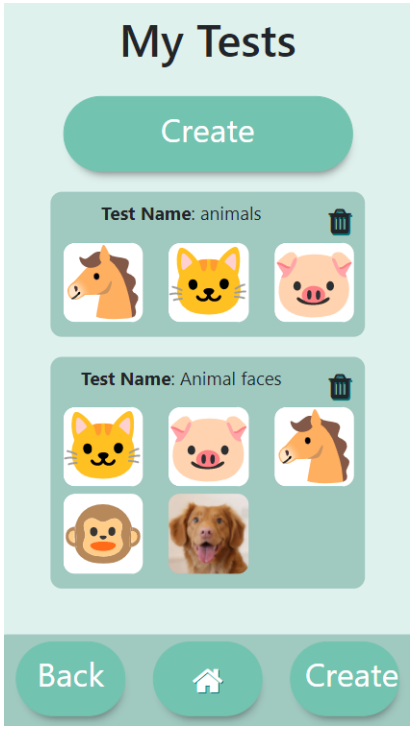


Figure 4.13: Test display view: main view of local test manager, on top and footer a create button to access Test Creator. In the center of the screen all local tests are displayed and has an option to delete it.

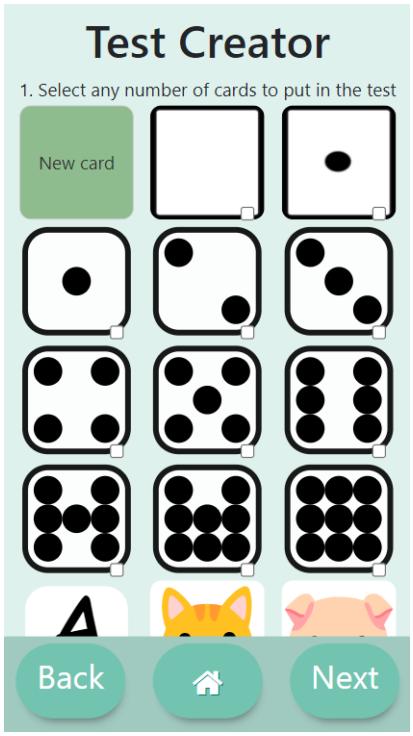


Figure 4.14: Test Creator view: First step view for Test Creator. New cards can be created from this view or select one or more cards to create a new test.

internal process to request a sound from the API with the related text for each one of the cards that declared a text, and it might takes time so we show on screen a waiting spinner while this happens. When the creation of sounds is completed, we update the local storage adding a new Test to local Tests. A step by step example is described in Annexed B.6.

Logs

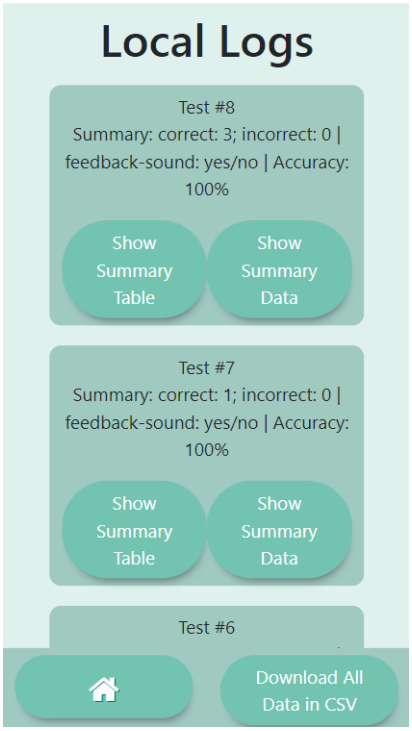


Figure 4.15: Logs main view: we can see session logs from last to first in top, each one with a brief summary and two buttons to extended summaries.

Session logs are one of the most important features of InCA BAT, giving a powerful tool to researchers to analyze data from Blind Associative Tests. Because of that, we carefully defined all the needed information with Cunha, based on her own research methodology, saving information from a session in real time, and storing it once the test concluded. All the needed data to save from each BAT session is described in Section 3.3.2.

To retrieve session Logs, we implemented a view to display all created logs in user browser, see Figure 4.15. In this view, we added 4 features to see logs in different level details, the first is the display of all logs with test session number and a brief summary with accuracy, chosen feedback sound and number of correct and incorrect selections (Figure 4.15). Second feature is a human readable log summary, displayed when clicking on *Show Summary Data*, showing one line with date, test name, repetition number, result, learner name, chosen card, cards on screen (ordered and with a * mark on correct choice) and time reaction for each selection on a test session, see Figure 4.16. Third feature is a table with the same information, Figure 4.17, and Last feature, but considered the most important, is a button to download all the logs data from all the sessions in a csv format file, with more columns that are described also in Section 3.3.2.

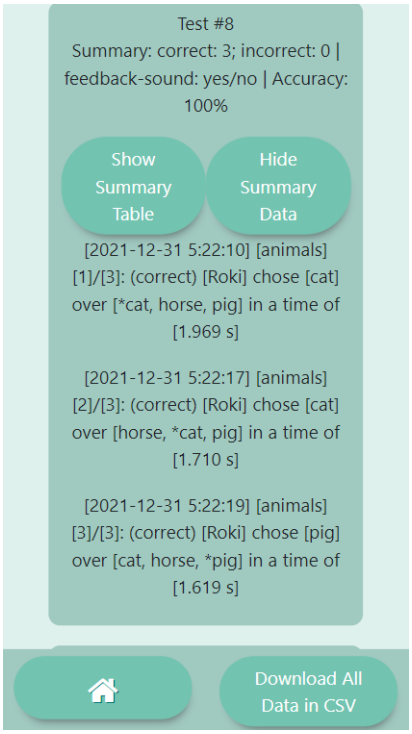
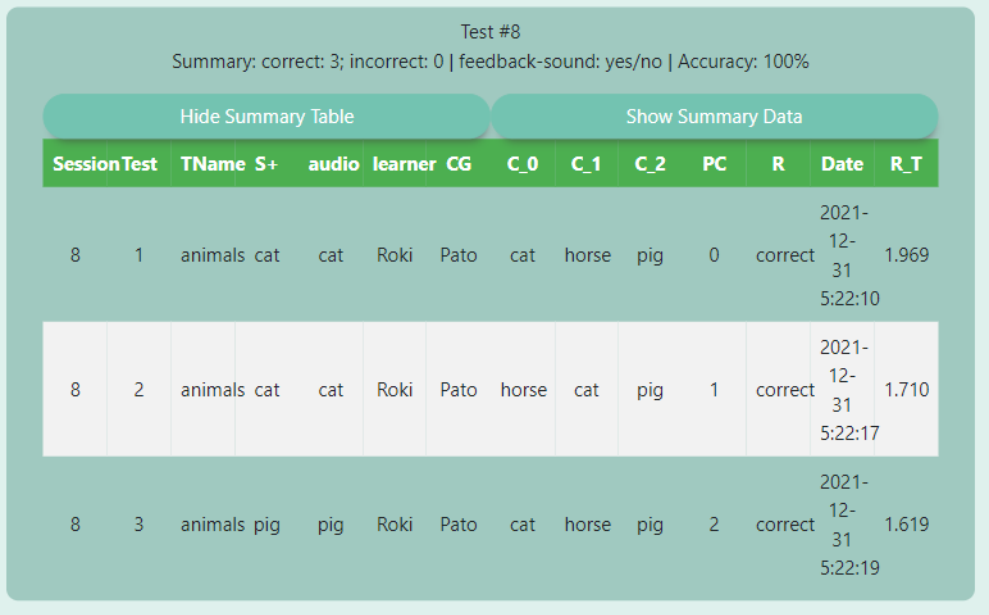


Figure 4.16: Logs view summary data: When clicking *Show Summary Data* button, it displays a list with all test repetitions in a session, in a human readable format.



Session	Test	TName	S+	audio	learner	CG	C_0	C_1	C_2	PC	R	Date	R_T
8	1	animals	cat	cat	Roki	Pato	cat	horse	pig	0	correct	2021-12-31 5:22:10	1.969
8	2	animals	cat	cat	Roki	Pato	horse	cat	pig	1	correct	2021-12-31 5:22:17	1.710
8	3	animals	pig	pig	Roki	Pato	cat	horse	pig	2	correct	2021-12-31 5:22:19	1.619

Figure 4.17: Logs view summary table: When clicking *Show Summary Table* button, it displays a list with all test repetitions in a session, in table format. This image was taken in a bigger resolution (tablet/iPad), because its not readable on small screens.

Test Selection

Test selection view can be accessed from main page clicking *Start Test*. In this view we can select a Test from a list and see a preview of the cards of the test. When we choose a Test from this view, it is changed globally and stored for the current session, it resets if we reload the web page.

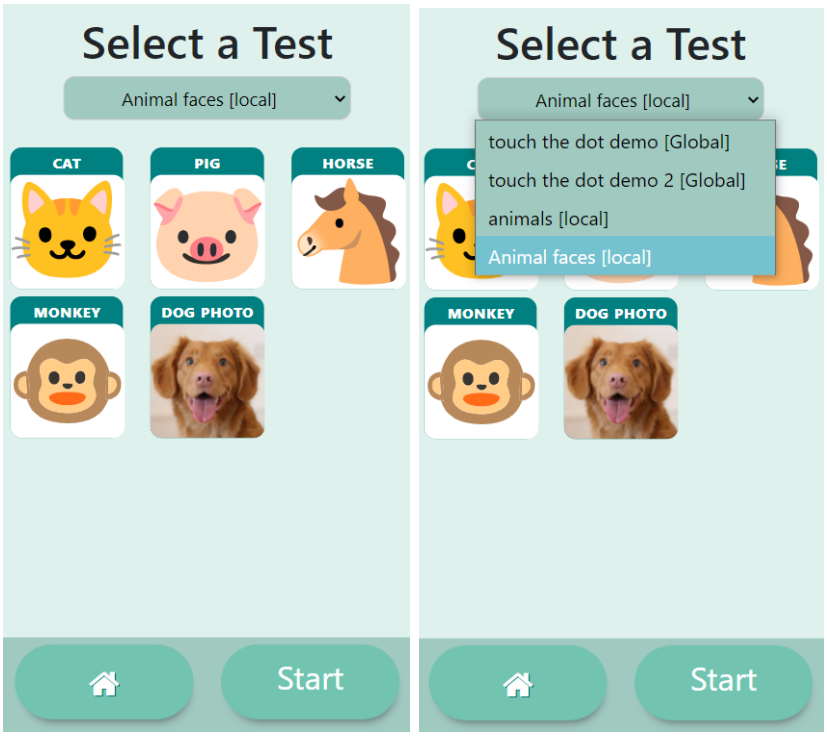


Figure 4.18: Test Selection view: In this view the selected test remains for all the session and can be selected a global or local test. When test is selected there is a preview of it, and if card image is clicked the card sound is displayed.

4.4 Backend

InCA BAT backend was build using Flask, without any user interface, just the base of an API connected to a non-relational database in mongodb. There is currently just one endpoint, that allows us to generate an audio file from text using IBM text to speech generator, with a limited free version. The database is deployed in mongodb Atlas and API in InCA server. Source code is stored in a the github repository `https://github.com/plt1994/inca-bat-backend`.

API and Database

The API structure is described in Figure 4.19. The application logic is located `app` directory: `apy.py` has the endpoints, `client.py` has the functions that interact with database and with **utils** contains major part of the program logic, `views` manage user requests to the API and interact with `client`. Database model is located in `database` directory, generated sounds are stored in `static/sounds` and environment settings in `settings.py` (connection to database and flask application settings).

The database is deployed in a free cluster in mongodb Atlas, that can be directly connected and used after a fast setup. We chose this option to avoid looking for a server to store a database in early steps, but it remained until the end of the project without changes. We suggest making a big update of backend to adapt to final frontend system.

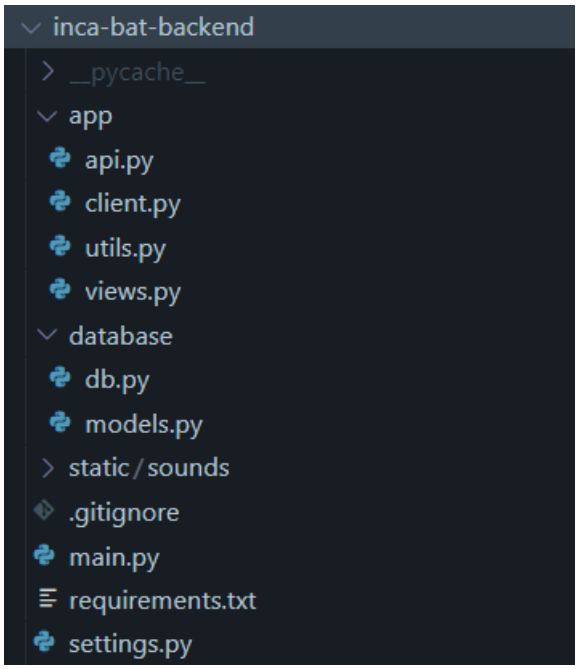


Figure 4.19: InCA BAT API structure based on traditional MVC architecture, from top to bottom: app directory contains views, endpoint urls (api.py), and utils (text to speech), database folder includes connection logic and models, static folder store generated files like sounds or images, and finally `main.py`, `settings.py` and requirements.txt include environment and deployment configurations.

Text to Speech

InCA BAT needed an option to associate an audio to a Card, but it was costly for the user to generate audios each time a test was created, especially if a test had many cards. Because of that, we decided to automatize this step, creating an audio from a text using a text to speech API from IBM. this service offered a free to use limited usage of 10000 letters per month, and we considered this limit big enough for our needs now and for a near future.

The process to obtain an audio from a text starts in *Test Creation* view in *frontend*, where the user set a text message on each card needed for a Test. When the test creation setup is sent, InCA BAT API calls a `text_to_speech` internal function that makes a request to IBM API for text to speech. The response of IBM API is an audio file in binary, that we save in the folder `static/sounds`, with a unique name, that can be retrieved from `https://buho.dcc.uchile.cl/inca-bat-api/static/sounds/<filename>`.

To avoid creation of repeated text to speech requests, and the same audio file with different names in the server, we save in database a document for each request with a new text that was not requested before, and if the text was requested, it does not create a file but return the last available file. It means that we only generate one audio for each text.

4.5 Deployment

A critical part of the application distribution is the deployment. So we explain here, what is the process we followed to deploy InCA BAT *frontend* and *backend*, on InCA servers. Starting for *frontend*, thanks to Svelte, the production environment building is simple and take 6 steps.

Step 1. Connect to InCA server by SSH:
`ssh inca-bat@buho.dcc.uchile.cl`

Step 2. Download InCA BAT repository (using git in this example):
`git clone https://github.com/plt1994/inca-bat`

Step 3. Open `inca-bat` folder:
`cd inca-bat`

Step 4. Install dependencies:
`npm install`

Step 5. Build Source Code:
`npm run build`

Step 6. Copy content of folder `public` to `/home/inca-bat/public_html`:
`cp -R public/* /home/inca-bat/public_html`

To deploy *backend* we have more options, but an easy one can be done using command `screen` and `pip`, from python. Following steps describes the process:

Step 1. Connect to InCA server by SSH:
`ssh inca-bat@buho.dcc.uchile.cl`

Step 2. Download InCA BAT backend repository (using git in this example):
`git clone https://github.com/plt1994/inca-bat-backend`

Step 3. Open `inca-bat-backend` folder:
`cd inca-bat-backend`

Step 4. Install requirements.txt:
`pip install -r requirements.txt`

Step 5. Create a screen:
`screen -S inca-bat-backend`

Step 6. Run server: `python3 main.py`

After this, the application will be available on `https://buho.dcc.uchile.cl/~inca-bat/` and backend will be available in `https://buho.dcc.uchile.cl/inca-bat-api` (to confirm if backend is available, access to `https://buho.dcc.uchile.cl/inca-bat-api/inca-bat`). It is crucial to know that if the server shutdowns or reboots, the current screen instance for the backend will also turn off, meaning that we need to start the application again following steps 1, 3, 5, and 6 (steps 2 and 4 are no longer necessary).

Chapter 5

Validation

The validation of InCA BAT required an active participation of the client, Cunha, and non-human animal learners, Ellie and Isabelle, two Cockatoos that lives with Cunha. We ensured that our project is ethically correct (see Section 5.1) and the first steps were directly related to validate that learners can use the Learner view. After critical validation of usability by non-human animals, we validated the usability of Blind Associative Tests, material creation and data collection with logs on informal interviews while gathering requirements, detailed in Section 5.2. In the last interview we made a demo of the application followed by a semi-structured interview with closed and open questions, validating all the important features, usability of the software, nearly future use and future work to improve the application.

5.1 Ethics

Non-human animal cognition research involves direct interaction with the subjects of study, in this case small non-human animals like parrots or mice. Because of that, in this project we were aware of this and ensured that our application will not cause any type of abuse from the human researcher to the non-human animal learner, directly or indirectly because of the use of the application while validating Learner views, and also a direct harm because of the application itself, such as stress, fear or any detected effect to the subject. In fact, InCA BAT might improve learners quality of life if combined with other applications like communication boards.

We ensured to protect the welfare, autonomy and dignity of each animal (human or non-human) in this project [7], by repeatedly testing all the interfaces before giving it to the client, and our client repeatedly tested the application before using it with any other animal that plays as a Learner on InCA BAT. No one was obliged or tried to oblige other user to use the application without their consent. As developers of the system, we ensured to give enough time to our client to test any feature involving a Learner any time it was needed.

Meeting Date	Meeting Details
October 19th 2021	First submit, Blind Associative Testing validation (Learner View) and feedback for new features.
October 26th 2021	Follow up, re-validation of fixes requested on first meeting and second submit, custom Test and Card creation. Also requirements gathering for logs visualization.
November 9th 2021	Third submit, Logs and data recollection. Revalidation of custom test creation after fixes.
November 16th 2021	Re-validation of Log fixes and finished requirements gathering for new features.
December 14th 2021	Final Submit with user interface enhancement and last software validation.

Table 5.1: Meeting details: We had 5 meetings at the end of each sprint, from October to December. Details of each meeting are described in second column.

5.2 Validation Process

In order to validate the developed software usability and its features, we counted with our client's validation, that is also user of the application, Jennifer Cunha. The validated points were 4: Blind Testing, Material creation, Data recollection and Software usability.

We had 5 meetings, the first 4 for requirements gathering and partial informal validations of implemented features, details in Table 5.1. The last validation of the software was realized in during the last part of the last meeting, in two parts: Evaluation from 1 to 5 ((1) Totally disagree, (2) disagree, (3) neutral, (4) agree, (5) completely agree) and open questions. Last evaluation questions and details are described in Annexed D.

5.3 Results

From 1 to 5, Cunha evaluated each one of the points presented on the last interview [2021-12-14 Tue], and replied 3 open questions. All the questions were explained to ensure that it was clear for the client when completing the evaluation. Results for open questions are described belows and results for closed questions are described in Table 5.2. Open questions:

1. **What do you think the app needs or what would you like to see in it.**
   (a) Uploading one's images easily
   (b) More feedback sounds (like "great" voice) and colors to make it more exciting/engaging for the bird.
   (c) Share tests across devices
   (d) Back-end to gather the statistics from various devices and facilitate their management.
   (e) Re-engaging feature (calling back after some time if no interaction) for the enrichment

2. **If there were not a log, would the digital set-up be better or worst than an analogical one?**
   It is a toss-up: Tablet makes the blind setup easier (than the cards) but (some) birds have (still) to learn to touch tablets.

3. **Do you use the application in one or many devices?**
   InCA-BAT was tested on 2 distinct tablets (there are 3 tablets use by the 4 different birds)

Question	Result
The application was easy to use	☆☆☆☆
The application was simple and straightforward to use	☆☆☆☆☆
The workflow of the application is intuitive	☆☆☆☆☆
It is easy to create an image card	☆☆☆☆
It is easy to create a text card	☆☆☆☆☆
It is easy to create an emoji card	☆☆☆☆☆
It is easy to create a test	☆☆☆☆☆
It is easy to delete a test	☆☆☆☆☆
The application allows the realize Blind Associative Tests	☆☆☆☆☆
The application allows the creation of a large variety of Blind Associative Tests	☆☆☆☆☆
The application allows to see logs in a useful way	☆☆☆☆☆
The application allows to export useful logs (human readable and csv)	☆☆☆☆☆
The logs will be useful for some research purpose	☆☆☆☆☆
I would like to use this application as a general user (trainer)	☆☆☆☆
I would like to use the application for investigation purpose	☆☆☆☆☆
I would like to use the application for investigation purpose (even if it did not have a log)	☆☆☆☆☆
The application has the potential to enrich the life of animals	☆☆☆☆

Table 5.2: Validation Results: The number of stars represent the level of agreement (☆) Totally disagree, (☆☆) Disagree, (☆☆☆) Neutral,(☆☆☆☆) Agree, (☆☆☆☆☆) Completely agree. The worst evaluated was *agree* and the best was *completely agree*

5.4 Analysis

In the last meeting and with the last validation interview, we evaluated the general usage of the application and specific requirements gathered in all the meetings with the client and also user of the software.

The validation covered three major points: successfully execute a blind associative testing as a choosing card video game, add new custom tests, and obtain complete logs from every game session. But also verifies the usability and the expected usages of the application, with open questions. The main results of the evaluation can be summarized in the following 4 points:

1. InCA BAT can easily create and delete custom material, with all the types of cards.

2. The client validated that with InCA BAT she can create and realize a large amount of blind associative tests. This is related with custom material creation and the application settings, allowing a Teacher to configure with many possibilities a Test and a Session.

3. InCA BAT Logs can be easily seen, are readable and understandable, easy to export and useful for research purposes.

4. The client would like to use this application for playing time and investigation, with or without the logs.

With all the points we validate that blind associative testing setups can be improved using InCA BAT, and also minimize the risk of bias due to the minimal interaction of the Teacher once the application started a new session, not letting choose the cards nor the questions in the whole session. Also, this application can be used to validate ACI hypothesis due to the complete logging system and easy data retrieve, giving enough data to the researcher and a reliable automatic way to obtain BAT sessions data.

Chapter 6

Conclusion

To conclude this report we summarize our contribution in Section 6.1, mentioning the importance of our work for ACI and specifically for our client, Jennifer Cunha. We discuss learned lessons from this project in Section 6.2, about obtained results, methodology of work and other topics useful for our future career and for future students who follow similar strategies for ACI projects, and finally we outline potential future developments and their relative cost in Section 6.3.

6.1 Contribution

We improved the way how a Blind Associative Tests applied to non-human animal cognition research can be done, designing and developing a web application for touch interfaces to emulate traditional Blind Testing with paper cardboard cards. Cunha stated, as a ACI researcher, that InCA BAT is now a core application of her daily routine. It is considered a reliable blind testing tool, that simplifies the creation of new cards and sessions, automatically randomize every test in a session and captures session logs on the fly. All the logs can be accessed from the main page of the application and are also downloadable.
InCA BAT is an open source code, available in two public repositories, one for the frontend using Svelte (javascript, html, css), in `https://github.com/plt1994/inca-bat` and one for an API using Flask (python 3, mongodb) in `https://github.com/plt1994/inca-bat-backend` partially coded but extensible, to add new functionalities. The last version is running in `https://buho.dcc.uchile.cl/~inca-bat/` and the backend in `https://buho.dcc.uchile.cl/inca-bat-api/inca-bat`

InCA-BAT was tested on 2 distinct tablets (there are 3 tablets used by the 4 different birds). In order to validate the usability and all functionalities of the software, we measured three important points.

1. Material generation: It is possible, simple and intuitive to create, delete and edit cards and tests.

2. Blind Testing: The non-human animal can use the application, it is, playing the game touching only one card each time without accidentally leaving the game, complete a game and possibly play again.

3. Log generation: The application collect all the data for a session on the fly, it is current Test setup, Settings, cards positions on the screen ordered from left to right, correct card, selected card name, selected position, card audio (represented in words), results, date and response time. The Tests logs can be retrieved to a file and seen on the application itself.

"I use the application a lot, it has become part of our core learning process" – Cunha, December 14th 2021

6.2 Discussion

Along the development of InCA BAT, there were low and high progress points, mostly due to methodology and personal situations organizing self work timing. We learned how powerful an agile methodology of work can be, allowing us to move from low to high productivity while obtaining more requirements in short time, just developing the most important parts in a project. This methodology helped us to release in half of the time a large amount of features, but it also made us refactor our project many times because of misunderstanding with original plan of the client, because of that we suggest to use flexible technologies when working with agile methodologies, for first steps or for all the project, designing and developing in no more than 2 weeks before validating state of the project with the client.

Meetings with the client are such a great opportunity for gathering requirements, but as a software engineer we need to take it carefully when accepting requirements in first steps of the software development, mentioning to the client that we are analyzing each request but prioritizing the most important parts for a MVP (Minimum Viable Product), and slowly add new requirements on each opportunity. We made some mistakes in first steps organizing meetings with the client, such as not clear time, date and place for meetings. When the client does not live in the same timezone, it is important to clarify the day of the week, the local time and date for each one of the guests and if possible what is the meeting about, to prepare ourselves but also for our client to understand the objectives of the meeting.

When gathering requirements, we recommend to ask as much as possible and make it a clear development piece, specially on first meetings. Assuming that everything has more difficulties as expected, and calculating also testing and deployment time when on each cycle of development. If we made an overestimation, we can always use our time to add extra features and test more our application, and this point is more relevant in ACI projects, because our users will be non-human animals or they will be directly related to the user, giving us an ethical responsibility to protect the welfare of non-humans, making us think and solve possible negative reactions such as stress or harmful situations, and this requires to test first all the non-human animal interfaces we created.

One of the most important parts of software design, is the final validation of our application. In many points of the developing of InCA BAT, we forgot the main objective of the project while designing interesting new features, so we recommend to have in mind all the time our final goal and ask ourselves (and also the client if they request non-viable features) if current development helps us to achieve this goal.

Finally, usability, reliability and extensibility needs to be present all the time when we develop a new piece in our software. Usability makes our software easy to use and to control for the users, so each time we design a button, interface, or choose a color or any detail, we need to think if this would be easy to use and understand for the final users. Reliability is also important, because users will believe in our software, and it needs to respond consistently and make what it says it can do, for example InCA BAT needs to give correct details for Logs, or the research of our users will be invalid. Extensibility does not affect the client direcly in short time, but if the software is successful and needs future changes, another software engineer must be able to add new features, or the software will keep outdated.

6.3 Future Developments

We implemented most of the functionalities needed by our client, but there are plenty of features and possible future work to do from this point. We describe three important possible future development and research, and we list many more work and features in Annexed E.

InCA BAT can create a large amount of different tests, mainly thanks to easy card creation, such as emoji and text cards, but adding other images are difficult, because the Teacher must provide the url of the picture, and cannot control the shape of each picture if added this way, just adjust the size. We propose as future work to extend the card creation process, allowing the user to add pictures from the used device or directly search recommended pictures from the application (such as WhatsApp gif searcher for chat). Such a feature will need to extend also the backend adding more endpoints and a CRUD (create, read, update and delete) for images, that can be extended to Test creation, and add users to manage client logs and test synchronized (manually or automatic). But needs to keep the application client based, for users having unstable connections. This work can be part of a undergraduate *Memoria de título* with medium difficulty that can be done by one student in 4 to 5 months.

Other needed future work is having an administration and analysis system, also called *admin system* or *backend*, where all data from InCA BAT frontend from each user can be accessed, included raw data and processed data, such as statistics of usage or other needed information useful for ACI researchers to validate animal cognition researches. This work can be done in 3 to 5 months, and requires to integrate current *frontend* to a *backend*, and extend it with current functionalities and database data.

Two possible interesting research works are *Card Randomization for Blind Associative Testing based on subject characteristics* and *Card hinting in BAT sessions and automatic difficulty tests adaptation*. We suggest these two topics for a master or doctoral thesis, due to the needed research with different subjects or even species and also needs to extend current

version of InCA BAT. More details on these two topics are available in Annexed E.

Bibliography

[1] Copy of 9.17.18 Isabelle baseline dot test 1. https://www.youtube.com/watch?v=-73o4sQwCLQ, Sep 2018. [Private video; Retrieved 2021-05-04].

[2] InCA-BlindComparativeTesting - REPL - Svelte - Barbay J. https://buho.dcc.uchile.cl/~inca-bct/, 2021. [Retrieved 2021-05-15].

[3] Jennifer Cunha [LinkedIn page] LinkedIn. https://www.linkedin.com/in/jennifer-cunha-75430b31/, 2021. [Retrieved 2021-05-04].

[4] Jennifer Cunha and Susan Clubb. Advancing communication with birds: Can they learn to read? 2018.

[5] International Conference on Animal Computer Interaction. About animal-computer-interaction. http://www.aciconf.org/aboutACI, 2021. [Retrieved 2021-05-04].

[6] J.D Jennifer Cunha and Dr. Susan Clubb. Advancing communaction with birds: Can they learn to read? In *Proceedings of the Conference "Animal Computer Interactión" (ACI)*, 2018.

[7] Clara Mancini. Towards an animal-centred ethics for animal–computer interaction. *International Journal of Human-Computer Studies*, page 98 pp. 221–233, 2017.

[8] Michael Trestman. Clever Hans, Alex the Parrot, and Kanzi: What can exceptional animal learning teach us about human cognitive evolution? *Academia.edu*, 2015.

ANNEXES

Annexed A

Requirements

A.1 Blind Testing Basic Setup and Learner View

- **Important**

  1. There is a Settings view that shows all the current values of all modifiable parameters and allows to modify it.
  2. Settings view has the following modifiable parameters: Number of repetitions, number of cards on screen, cards separation, feedback time delay in seconds, enabled text card, card height, card width, enable card audio message, feedback sound after click.
  3. The number of repetitions minimum value is 1 and has no maximum value. Default value is 5.
  4. The number of cards on screen minimum value is 1 and maximum value is 5. Default value is 2.
  5. The cards separation minimum value is 0% and maximum value is 100%. Default value is 50%
  6. The feedback time delay minimum value is 0 seconds and maximum is 3.5 seconds. Default value is 1.5 seconds.
  7. There is a test selection view
  8. There is a Learner View that has a big button with the word "Start!".
  9. In the test selection view: there is an option that allows to select only one test from a list of all available tests represented by the name of the test.
  10. In the test selection view: if no test was selected, the first time the application was opened, the default "touch the dot" test is selected.
  11. In the test selection view: there is a button to start the current test, that takes to the Learner View.
  12. In the Learner View: when button "Start!" is pressed, the selected test starts, showing N cards on the screen randomly chosen from the total amount of cards

of the selected test, shows the name of one card from the list of N cards on the screen and plays the audio for that one card. N is the minimum value between the number of cards of a test and the number of cards on screen shown on settings view.

13. In the Learner View, once the test started: Cards can be selected by clicking on it, on any part of the image of the card. Once the card is selected, if the played sound or text message on screen matches with the card, it plays a feedback sound. Feedback sound played is the feedback sound selected on settings view.
14. After the feedback sound is played, if the number of tests in a session is less than R, cards disappear from screen for N seconds, where N is the feedback time delay parameter from settings view. After N seconds the test starts again. R is the value of number of repetitions shown in the settings view.
15. After the feedback sound is played, if the number of tests in a session is equal to R, cards disappear from screen and session finishes showing a summary of the session correct and incorrect answer and a button to go back to test selection view. R is the value of number of repetitions shown in the settings view.
16. There is a button to restart the game session after it finishes, big enough to be clicked by the Learner or the Teacher. When it is clicked, creates a new session with the last selected test from the test selection view.

- **Normal**

1. In the test selection view: when clicking a card image from the preview, it plays the sound of the card if it exists.
2. In the test selection view: show the current selected test cards with their names in uppercase.
3. The current test keeps selected even changing views or after the game finishes. It will reset if application is restarted or it will change if other test is selected from the test selection view.
4. Settings view has the following modifiable parameters: Number of repetitions, number of cards on screen, cards separation, Learner screen background color, card background color, foreground color, feedback time delay in seconds, time exit holding time, enabled text card, card height, card width, enable card audio message, feedback sound after click.

Annexed B

InCA BAT interfaces

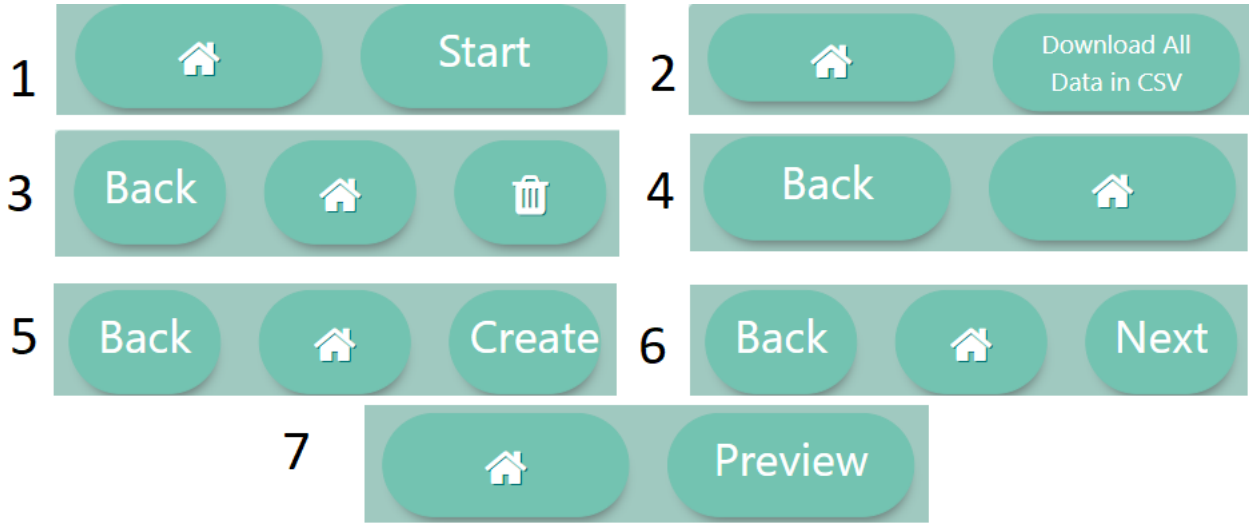


Figure B.1: Teacher views footers. All possible footer button structures for 1 to 7: 1) Test Selection. 2) Local Logs. 3) My Cards display. 4) Cards creator. 5) My Tests display 6) Test creator. 7) Settings.

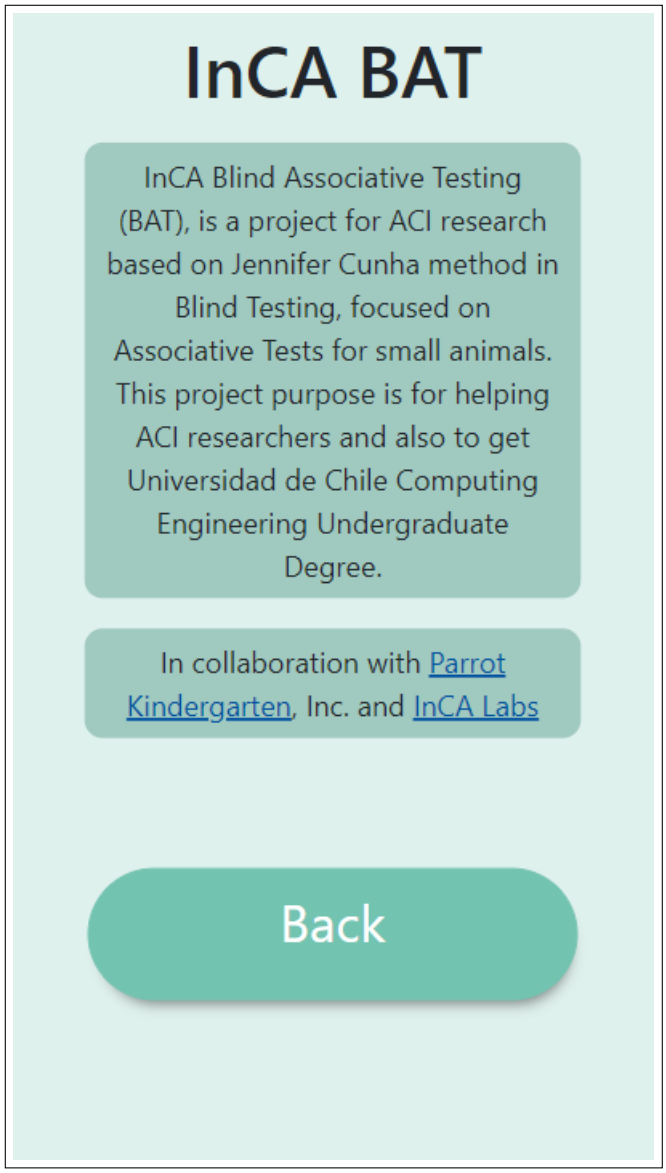


Figure B.2: InCA BAT About view: includes a description of the project and mentions to Parrot Kindergarten and InCA Labs.

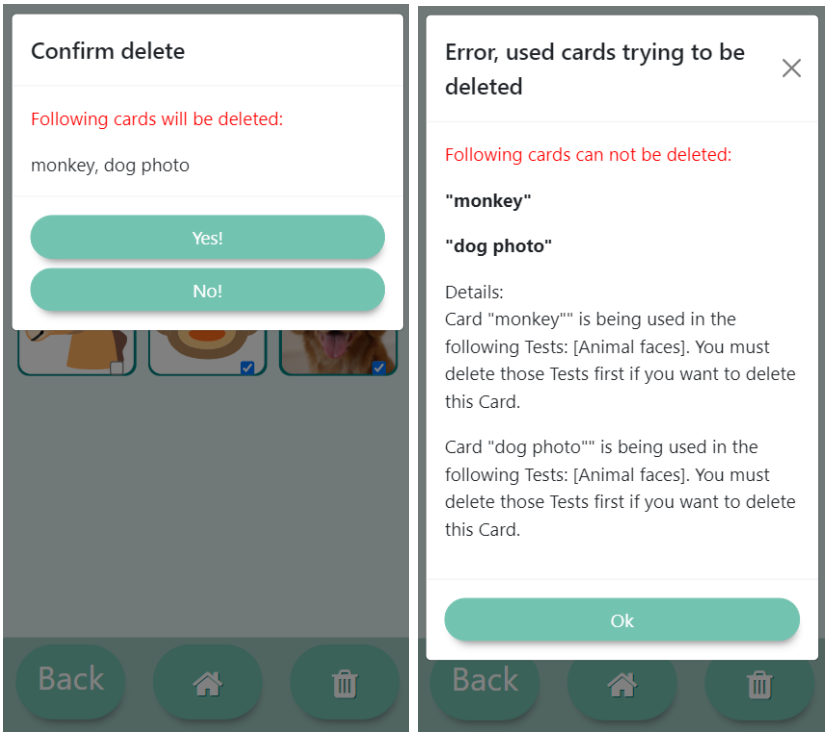


Figure B.3: Card delete examples: On left a confirmation message to delete without restrictions. On right a warning that does not allow to delete pictures and its detail.

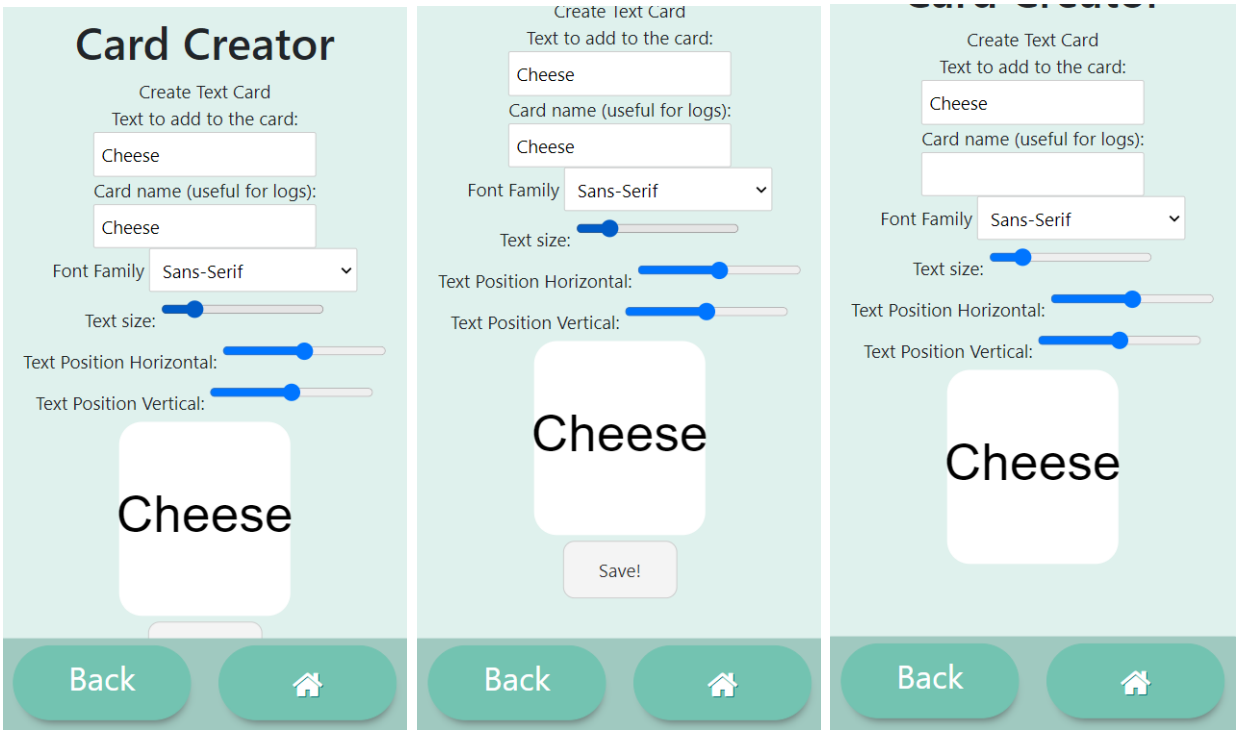


Figure B.4: Card Creator: Example for text creator, on left configuring a Cheese word with text size and position. On center same image but scrolling down to see *save* button. On right, cheese card added already, removing button for save.

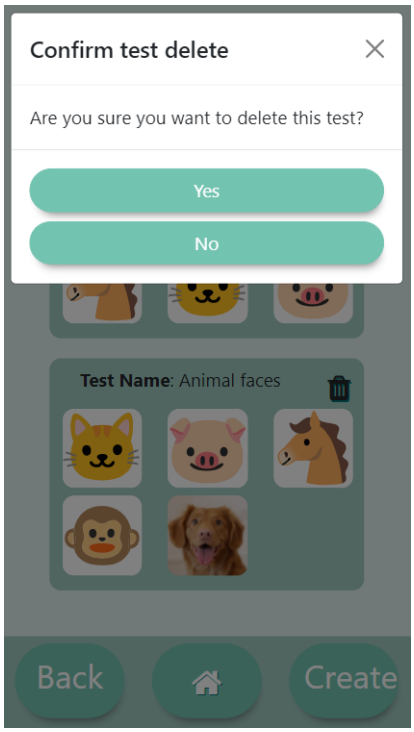


Figure B.5: Test deletion example when clicking the icon of trash can requesting confirmation.

Figure B.6: Test creation example: From 1 to 9 starting from top left, moving to right, and finishing in bottom right. 1) Tests display, click on Create. 2) and 3) Test creator, choose 5 animal cards. 4) set a text to each animal card. 5) set a name for the test. 6) confirm creation. 7) creation in progress. 8) creation finished, click on finish. 9) Final result with our new test.

Annexed C

Source Code Structure

C.1 Frontend Structure

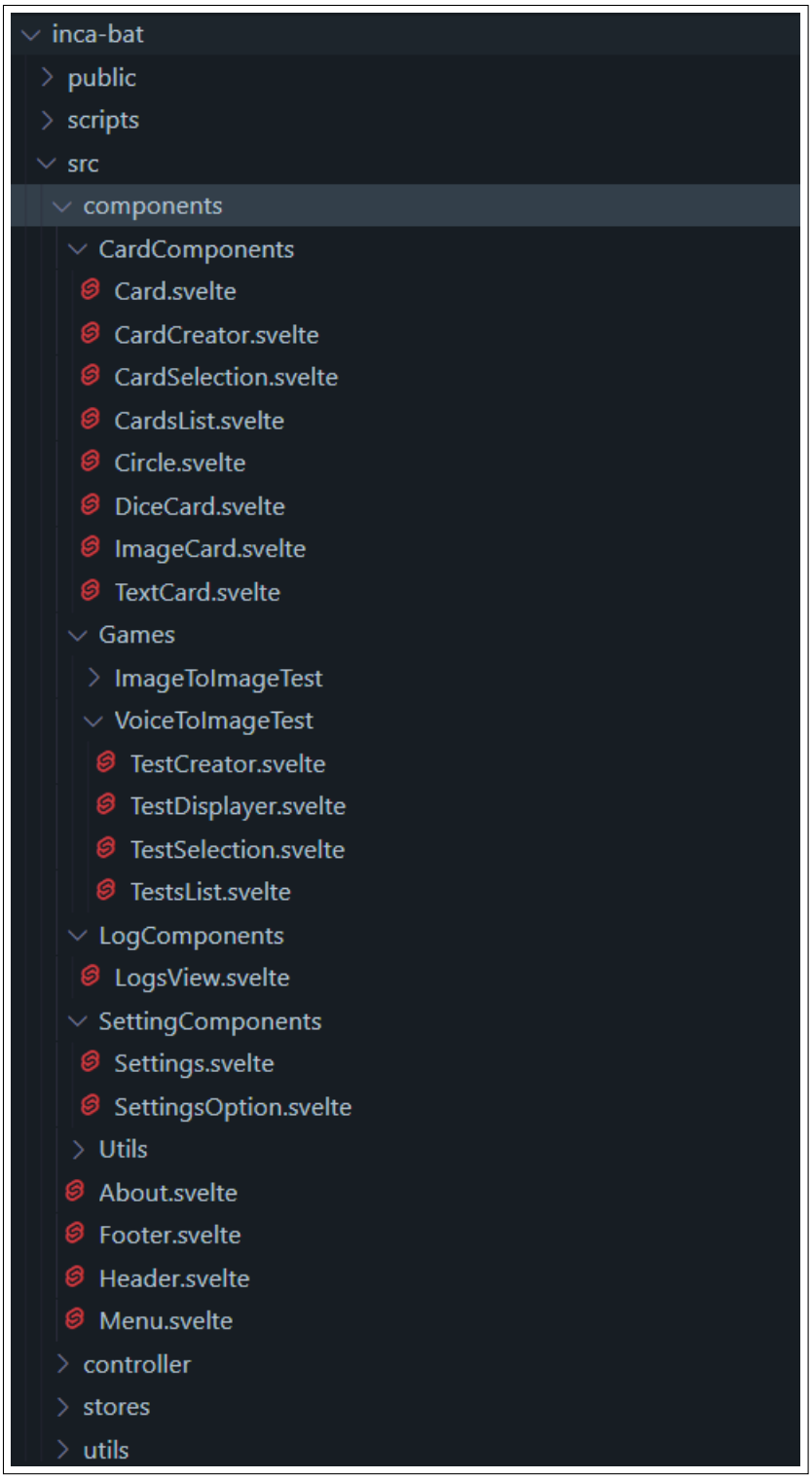


Figure C.1: Source code frontend structure with extended file view. All components for views are visible.

Annexed D

Evaluation

D.1 Closed questions

- Usability validation of interfaces

  1. The application was easy to use
  2. The application was simple and straightforward to use.
  3. The workflow of the application is intuitive
  4. It is easy to create an image card
  5. It is easy to create a text card
  6. It is easy to create an emoji card
  7. It is easy to create a test
  8. It is easy to delete a test

- Blind Associative Testing and Logs

  9. The application allows the realize Blind Associative Tests.
  10. The application allows the creation of a large variety of Blind Associative Tests
  11. The application allows to see logs in a useful way.
  12. The application allows to export useful logs (human readable and csv).
  13. The logs will be useful for some research purpose.

- How the application will be used

  14. I would like to use this application as a general user (trainer).
  15. I would like to use the application for investigation purpose.
  16. I would like to use the application for investigation purpose (even if it did not have a log).
  17. The application has the potential to enrich the life of animals.

D.2 Open questions

1. What do you think the app needs or what would you like to see in it.
2. If there were not a log, would the digital set-up be better or worst than an analogical one?
3. Do you use the application in one or many devices?

Annexed E

Future work list

1. Important Features:
   (a) **Learner View Cards display**: Improve the responsiveness of card displaying on different screen sizes and refreshing card displaying when the screen is rotated.[Normal feature to include in another plan, such as backend integration]
   (b) **Upload Images for Card Creation view**: For now all the cards for any test can be created by adding the source link of an image while creating a new image, or it creates on the fly an SVG code based on the card details. It is required to extend card creation to directly upload a picture or photo from the user device, adjusting it after upload to ease the editing for the user. [Normal feature to include in another plan, such as backend integration]
   (c) **Offline loading user data**: A first approach to understand how data is allocated on client side is adding a feature to download the Local Storage on a file from browser client, to upload it later in other client session or device, to duplicate all the settings, local tests, cards and logs created in one session. The logical next step is adding this same feature without the need of downloading a file, but using a log-in system to keep data among sessions (next point).[Simple feature to include in another plan]
   (d) **Backend Users and CRUD for tests and cards**: Add users and log-in system, with the possibility to save user data on database to create, retrieve, edit and delete tests in any device connected with the same user account.
   (e) Extend Local App with Online Tests and Cards
   (f) Create a complete user administration view for data analysis, where all the logs of current user can be displayed and downloaded, create, edit and delete new tests and cards, general overview for each registered Learner and its progress for each test type such as learned cards and audios, best tests and cards, worst tests and cards and other useful data to analyze for non-human animal cognitive research. [Memoria-full period]
   (g) **Audio repetition for current test**: After few seconds of inactivity, replay the sound or give the option to the Teacher to replay the sound. [Simple feature to include in another plan]

(h) **Safety checks for card and test creation**. [Simple feature to include in another plan]

2. Useful Features:

   (a) **Online sharing system**: There is a big potential for the application to be used by many people, each of them can create new cards and tests that measure specific skills and they would like to share these tests with other users. This might also be a feature for *administrators* of the system, to add more *Global* tests that can be used in any device by anyone that uses the application.

   (b) **Not Online dependency**: There might be cases where there is no good connection in the moment to start a session, so all the online dependencies will not work and the tests will be incomplete. To solve this, we can add a feature to download locally all needed data to run tests to use without connection in future sessions. Note that these is not considering the possibility to create new tests, due to the API creates audio files, but text cards can be created.

3. Research Features (all possible for master or undergraduate thesis):

   (a) **Multiple solutions for same audio on a Test**: Study and evaluate if its needed to add a feature where a test has multiple valid solutions for same audio. For example: We created a Test to evaluate if the Learner can distinguish animals whether is shown on emoticon, drawing or photo. We start the session for that test and we have three cards on screen: two of them are horse represented with different images and the other is a cow. The audio says "where is horse?". How can this feature affect the Learner on its learning process?

   (b) **Multiple audios for same card**: Study and evaluate if its needed to add a feature where a test has cards with multiple questions with the same solution, it is having different audios for the same card. For example: We created a Test to teach new graphemes to the Learner, specifically letters from the English alphabet. We have 1 card with the letter **A** with two possible sounds: *apple*, *alphabet*. Starting a new session with such test only display one card on screen, but the audio is randomly selected. How can this feature affect the Learner on its learning process?

   (c) **Image to Image or Sound to Sound Tests**: Study and evaluate possible solutions to add Image to Image association Tests or Sound to Sound Association Tests. How can these tests be used and what kind of non-human animal cognitive researches can be done?

   (d) **Card Randomization for Blind Associative Testing based on subject characteristics**: Study and evaluate cards randomization on Learner view, for different situations like learning new cards, reinforcement of most failed cards, discard of most failed cards, discard of most accurate cards and other situations worth to be studied. Depending on subject characteristics, this randomization might need to be adjusted, because of different capabilities of each subject.

   (e) **Card hinting in BAT sessions and automatic difficulty tests adaptation**: Study and evaluate how card highlighting with colors on correct cards can help the Learner to improve the learning of new cards, and how based on the progress

the highlight will change the opacity. Also this research might include other reinforcement systems such as audio reinforcement or automatic reinforcement tests creation with most failed cards on last test or last period.

(f) **Incremental or decreasing test difficulty**: Study and evaluate possible implementations for incremental difficulty of tests when there is a good progress learning an specific test, or decreasing the difficulty if there are many errors in test.

Annexed F

Diagrams

F.1 InCA BAT Navigation Map

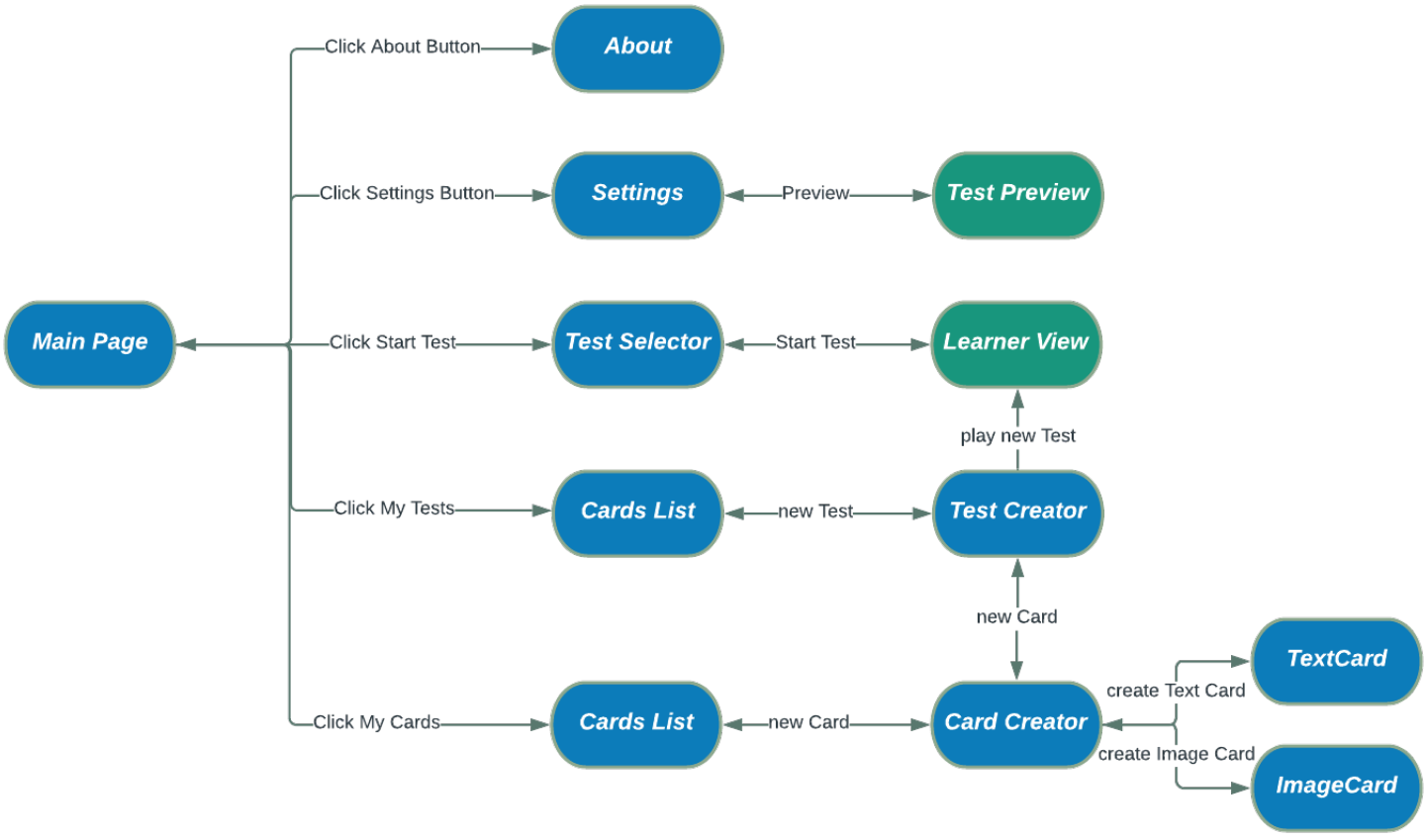


Figure F.1: InCA BAT Navigation Map: Simplified navigation map for InCA BAT. Teacher interfaces on blue, Learner Interfaces on green.