UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

DEVELOPMENT OF AN EDUCATIONAL ANDROID APP FOR CHILDREN WITH COMMUNICATION DIFFICULTIES

ZHIWEI LIN

Thesis supervisor
GRIGORI ASTRAKHARCHIK (Department of Physics)

Degree
Bachelor's Degree in Informatics Engineering (Software Engineering)

Bachelor's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Abstract

The objective of this project is to develop an educational Android application for children with communication difficulties, aiming to help them acquire knowledge and enhance their abilities and skills. The app offers an engaging learning experience by combining entertaining videos with interactive matching games. These activities support their daily lives and contribute to improving their overall quality of life.

Resum

L'objectiu d'aquest projecte és desenvolupar una aplicació educativa per a Android destinada a nens amb dificultats de comunicació, amb la finalitat d'ajudar-los a adquirir coneixements i millorar les seves habilitats i capacitats. L'aplicació ofereix una experiència d'aprenentatge atractiva combinant vídeos entretinguts amb jocs interactius de correspondència. Aquestes activitats donen suport a la seva vida diària i contribueixen a millorar la seva qualitat de vida.

Resumen

El objetivo de este proyecto es desarrollar una aplicación educativa para Android destinada a niños con dificultades de comunicación, con el fin de ayudarles a adquirir conocimientos y mejorar sus habilidades y capacidades. La aplicación ofrece una experiencia de aprendizaje atractiva combinando vídeos entretenidos con juegos interactivos de correspondencia. Estas actividades apoyan su vida diaria y contribuyen a mejorar su calidad de vida.

Acknowledgments

Firstly, I am deeply thankful to my project supervisor, Professor Grigori Astrakharchik, for his guidance, support, and constructive feedback, which were crucial to completing this project.

Secondly, I extend my sincere appreciation to Melania Chistè from TimeAut, a cooperative specializing in psycho-educational interventions for children and adolescents with developmental disorders, for her opinions and guidance. I also wish to acknowledge "Il Quadrifoglio" - Socio-Rehabilitative Center for Developmental Age of ANFFAS TRENTINO ONLUS for their contributions.

Finally, I want to express my gratitude to my family and friends for their support, understanding, and encouragement, which motivated me to complete this project.

Contents

List of Figures

List of Tables

Chapter 1

Context and Scope

1.1 Introduction

The main goal of the project is to develop an educational Android app that provides an engaging and educational experience for children, especially those with communication difficulties. The app will integrate with cartoon or child-appropriate video playback, periodically pausing the video to present educational tasks.

We decided to develop this app because there is currently no app on the market that offers similar functionalities, provides an interactive learning experience, and makes the entire process much more engaging and fun. This gap motivated us to create the app.

1.2 Context

This is a bachelor's thesis for the Informatics Engineering Degree, specialization in Software Engineering, at the Barcelona School of Informatics (FIB) of the Universitat Politècnica de Catalunya (UPC) - BarcelonaTech.

This thesis corresponds to Modality A [1], which means that the project is associated with the departments of the UPC. It is directed and supervised by Professor Grigori Astrakharchik from the Department of Physics.

1.2.1 Concepts

For a better understanding of this project, this section defines and explains some essential concepts needed for the development of the app.

Ahead-of-time Compilation (AOT)

Ahead of Time Compilation (AOT) is the process of transforming a high-level programming language into a low-level machine-readable language at build time. This makes the program perform better during runtime since there is no need for a real-time code translation.

Software Development Kit (SDK)

A software development kit (SDK), also known as a devkit, is a set of software-building tools bundled in a single package. Provides developers with easy access to all necessary resources in one download. An SDK normally includes essential tools such as a compiler, debugger, and often a framework or set of code libraries, making it faster and simpler to build applications.

Android App Bundle (AAB)

An Android App Bundle (AAB) [2] is a standard file format established by Google to replace the previous APK format, which was used to publish and distribute Android applications on the Google Play Store. Compared to APK, this format not only improves and optimizes resource usage but also offers additional benefits.

Picture communication symbols(PCS)

Picture communication symbols [3] are a set of colors or black & white visual icons, as shown in Figure 1.1, which are used to aid communication, especially for people with speech or language difficulties. These images or symbols are widely used in augmentative and alternative communication (AAC) to help people with communication challenges understand and express themselves.

Augmentative and alternative communication (AAC)

Augmentative and alternative communication [4] is a method designed to help people who have difficulty with verbal communication. This approach allows people to express themselves without relying on spoken language, using various tools and techniques that enhance or replace verbal expression.

Figure 1.1: Example of Picture Communication Symbol (PCS) [5].

1.2.2 Problem to Be Solved

Apparently, for children with communication problems or difficulties, there are not many educational resources or tools available that effectively combine learning with entertainment. Moreover, most of the tools on the market are primarily designed for use on computers, making them less convenient and portable compared to devices such as smartphones or tablets.

In addition, many technologies on the market focus on passive learning, where children primarily acquire information and communication skills by watching videos without participating in any interactive activities. As a result, these tools are less likely to capture children's attention during the learning process, which can reduce the overall effectiveness of improving their communication skills.

Moreover, it is generally considered a negative aspect to provide videos that attract children's attention, as they may become overly addicted to the screen and the content provided. However, in our case, we take advantage of this strong attraction for educational purposes, making the learning process more acceptable for the children.

In conclusion, our application can provide an active learning experience by integrating with locally stored cartoons or child-appropriate videos that periodically pause to present educational tasks, in this case, a matching game. This feature enhances children's engagement and makes the learning process much more effective and enjoyable for children.

1.2.3 Stakeholders

In the following, we have a list of stakeholders that are involved in the project.

- **Software developer:** This Android application would be developed by one developer, in this case the author of the thesis, and would be responsible for the entire project.
- **Children with communication difficulties:** They are the main users of this application since it is specifically designed to meet their educational and communication needs.
- **Parents, caregivers, and family members:** They are key stakeholders because they will assist and supervise the children while using the app, and they can also provide valuable feedback.
- **Schools and special education institutions:** Institutions or centers that integrate the app into their teaching methodology or therapeutic sessions to offer a more interactive learning experience for children with communication difficulties.
- **The final thesis supervisor:** Professor Grigori Astrakharchik plays an important role in guiding the author and supervising the development progress to ensure the final result of the project.

1.3 Justification

When considering the existing solutions developed to improve the communication skills of children with communication difficulties, there are several applications on the market, such as **Proloquo2Go** [6] and **CoughDrop** [7]. However, both only provide augmentative and alternative communication (AAC) tools, which basically help children communicate through visual symbols. These solutions are mainly focused on real-time communication rather than teaching and learning communication skills.

Moreover, apps like **Autism iHelp-Play** [8] are only available for iOS devices, and they do not include Picture Communication Symbols (PCS) in their application. Additionally, this app does not combine video playback with educational games, making the learning process less engaging and enjoyable. Furthermore, some applications, such as **YouTube Kids**, only play videos without providing any additional games or tasks to improve communication skills.

As a result, there is a need for a new solution that combines video-based entertainment with matching games. Professor Grigori Astrakharchik proposed this idea through the Thesis Offers Form, suggesting the development of an Android application that meets this general requirement. Our project offers a unique combination of these features, creating a

more enjoyable experience and providing tools to improve children's abilities, knowledge, and communication skills, thereby filling a gap in existing solutions.

1.4 Scope

1.4.1 Objective and Sub-objectives of the Project

The main objective of this project is to develop an educational Android application for children with communication difficulties, combining learning with entertainment. The app will feature video playback that periodically pauses to present educational matching games using PCS, text, images, or voice prompts, all designed to improve their knowledge and communication skills.

The secondary objectives of this project are to create an engaging and enjoyable learning environment by integrating matching games into the video playback, encouraging active participation. In addition, this project aims to design a user-friendly interface that allows both children and their parents or guardians to easily navigate and access all features and functionalities. Finally, the application will be developed and designed to work without an internet connection, making it accessible anytime and anywhere.

1.4.2 Requirements

This section provides an overview of the functional and non-functional requirements that the project must meet to ensure its successful implementation.

Functional Requirements

- **Offline mode:** The application must be fully functional without an internet connection, ensuring the app can be used in an environment where internet access is unavailable.
- **Personalized educational games:** The application must allow parents or supervisors to create personalized matching games with different types of PCS cards, images, text, and voice prompts based on each child's unique needs. They should also be able to choose which games to present and which not to and specify the number of games that need to be completed before allowing the child to watch a new video.
- **Interval configuration:** The app has to give parents or educators the option to adjust how frequently the video pauses to introduce games, allowing intervals like every 5 minutes, for example.

- **Presentation of an AAC system:** The application must provide a panel with cards presenting various actions in daily life, and each card is capable of producing sound, helping users to express their ideas.
- **User authentication:** The app must provide an authentication method to allow users to create accounts and log in. This ensures that only authorized users can access specific parts of the system and enables personalized experiences for each user.
- **Presentation of interactive game:** During the video pause, the application must present interactive and educational matching games that encourage the user's participation and enthusiasm for learning.
- **Progress tracking:** Parents or guardians must be able to track the child's progress through tables or charts and general statistical information such as accuracy, average time per game, etc.
- **Data export and import:** The application must enable data export and import, allowing information to be transferred to another device in case of changing devices.
- **Video Playback:** The application must be able to present and play child-appropriate or educational videos stored locally on the device, pausing at a specific moment to present tasks that need to be completed.
- **Language option:** The application must provide language options, allowing users to select their preferred language from the available choices to use the app.
- **User account management:** The app must allow users to change their password, edit their profile, log out, and delete their account.
- **Game difficulty configurations:** The app must allow parents or caregivers to adjust game difficulty based on the target user's performance. This customization ensures a more personalized user experience adapted to the needs and abilities of the player.

Non Functional Requirements

- **Accessibility:** The app should be accessible to children with communication difficulties, using clear icons or images.
- **Compatibility:** The app must be compatible with a variety of Android devices, such as tablets and smartphones, and need to adapt to different screen sizes without compromising the user experience.
- **Performance:** The app needs to be efficient, with fast responses to user interactions.
- **Reliability:** The app should be reliable and stable, avoiding crashes or bugs that could significantly affect the user experience.
- **Security:** The application must ensure data security, especially regarding the children's progress data.

- **Usability:** The app must be easy to use for both children and caregivers, with a child-friendly user interface. Buttons, icons, and navigational elements should be large, clear, and easy to see.

1.4.3 Potential Obstacles and Risks

During the development of the project, it is crucial to anticipate potential obstacles or risks that could arise and need to be carefully considered and managed to ensure the completion of the project.

Deadline of the project: This is an essential aspect to consider, as the final thesis has a fixed deadline. Proper planning and time management are crucial to ensure that all tasks are completed within the given time frame.

Inexperience with certain tools: The development process may involve technologies or tools with which the author of the project is unfamiliar. This learning process could slow progress and extend the development timeline.

Bugs: It is normal to encounter bugs during the software development process. Although some bugs may be resolved easily, others can be more complex and time-consuming and can significantly delay our development progress.

Technological failures: Software or hardware failure, such as a computer breakdown, can significantly affect development progress. These incidents not only cause delays but may also result in additional financial costs for repairs or replacements.

1.5 Methodology and Rigor

1.5.1 Working Methodology

The working methodology for this project is the Kanban methodology [9], as there is only one developer and there is no need to meet with other developers. Kanban methodology allows the management of the task in a way that can identify the status of each task, making it easier to track the current progress.

Kanban methodology enables workflow management without the need for iterations or sprints as required by other agile methodologies, such as the Scrum methodology [10], making it ideal for the continuous delivery of small adjustments or improvements during the development process. It is a highly flexible working methodology that allows developers to adapt to changes or new requirements easily.

In this case, I use Taiga to manage tasks, representing each task by a card. The cards will

be classified on a board with four different columns:

- **Ready**: This column contains all tasks that have been defined but not yet started.
- **In progress**: This represents tasks that have begun but are not finished.
- **Ready for test**: This column contains the tasks that have been developed but still need to be tested.
- **Closed**: This represents all the tasks that have been completed and tested.

1.5.2 Validation

GitHub: [11] We use the GitHub repository for version control, allowing us to access the code from different devices and recover previous versions of code if needed. This also works as a backup for our code.

Taiga: [12] This is the project management tool that helps us to manage tasks and monitor our progress by using Kanban.

For the validation, we will have a weekly meeting via Google Meet or face-to-face with the thesis supervisor, Grigori Astrakharchik, to validate tasks, report our progress, receive feedback, etc. For urgent situations or questions, we will use email for communication, and we will schedule additional meetings if it is necessary.

Chapter 2

Temporal Planning

2.1 Description of Tasks

The project is estimated to take approximately six months to complete. It started on 22 July 2024 and is expected to conclude in early January 2025. The estimated time commitment of this project is around 30 hours per week, although it may vary due to external factors.

In the following, we present a detailed description of our project plan and explain the tasks that need to be completed. These tasks are structured into three main parts: project management, project development, and project documentation.

2.1.1 Project Management

This section lists all the tasks needed for project management. For this phase, a computer with its accessories and the Overleaf [13] LaTeX editor are required.

- **PM1 - Context and Scope:** This task involves defining the context and scope of the project, which includes the methodology, stakeholders, problem to be solved, key concepts, etc. It requires 30 hours of work.
- **PM2 - Project planning:** This task focuses on developing the thesis plan, including the Gantt chart and risk management strategies. It requires 20 hours of work and depends on the completion of PM1 and GanttProject [14].
- **PM3 - Budget and sustainability:** This task involves creating a budget plan and conducting a sustainability analysis for the project. It also requires 20 hours of work and depends on the completion of PM2.

- **PM4 - Integration in final document:** The final document includes the content from PM1, PM2, and PM3, with all errors corrected and feedback incorporated. This task requires the completion of all previous PMs and approximately 15 hours of work.
- **PM5 - Meeting:** There will be meetings with the supervisor of the project every week via Google Meet for an hour to ensure the progress and the final result of the project.

2.1.2 Project Development

This section includes all tasks related to development, including preparation, learning process, development, and review. For this phase, a computer with its accessories, Visual Studio Code [15], and Android Studio [16] are required.

- **PD1 - Preparation of the programming environment:** Before starting the development of the application, it is necessary to configure and set up the programming environment for our computer. This includes installing the Flutter framework and configuring Android Studio to run the Android emulator. This task required 5 hours of work.
- **PD2 - Learn to work with Flutter**: Before starting the development process, it is essential to learn how the Flutter framework works. This includes not just watching tutorials but also applying the knowledge through hands-on practice; therefore, it is important that PD1 is completed beforehand. The estimated time for this task is around 25 hours, though consulting Flutter documentation may be necessary throughout the development process.
- **PD3 - Database design:** This task involves designing the database to support the application's data requirements. The database needs to include all necessary tables and attributes, following proper database design principles. This task requires approximately 5 hours of work and will utilize draw.io [17] to design a UML diagram.
- **PD4 - User interface design:** This task involves creating and designing various mockups to represent a user-friendly interface, providing a clear visual representation of the application. The estimated duration for this task is 20 hours, with Figma [18] being used for the design process.
- **PD5 - Database implementation:** Based on the database design from PD3, the database will be implemented using the sqflite package from Flutter. This step involves setting up all required tables, attributes, and database configuration. The estimated time for completion is 15 hours and depends on PD2 and PD3.
- **PD6 - Data access operations:** In this phase, all necessary data access operations, such as CRUD functions (create, read, update, and delete), will be implemented by Flutter. This step requires the completion of PD5. The work is estimated to take

about 45 hours, with potential adjustments as new requirements emerge during development.

- **PD7 - Game management:** This task focuses on developing the user interface for games based on the mockups created in PD4, alongside implementing key features such as card matching, drag-and-drop actions, and more. The estimated time for this is approximately 20 hours, and it also depends on PD6.
- **PD8 - User management:** In this task, user functionality will be developed, including login, sign up, log out, edit profile, profile management (view, edit, change password, statistics), and more. Both the frontend and backend will be built using Flutter. Additionally, user statistics will be presented using charts, tables, and other visual elements. This task requires the completion of PD4 and PD6. The estimated time is around 45 hours.
- **PD9 - Admin functionality of the game:** Here, the logical part and UI for admin account functionality will be developed. This includes managing games and cards, along with performing CRUD operations. The task is estimated to require approximately 45 hours of work and depends on the completion of PD4, PD6, and PD8.
- **PD10 - Admin functionality of AAC and data export/import:** In this phase, another admin functionality will be developed, including managing the AAC system and handling data import/export. This task requires approximately 40 hours of work and depends on the completion of PD4, PD6, and PD8.
- **PD11 - Video playback:** This task includes the creation of video playback features, allowing users to play, pause, and search for locally stored videos. The estimated time is around 25 hours and depends on PD4.
- **PD12 - Timer:** This part focuses on implementing a timer that will be in charge of pausing the video to display educational games. In addition, it will allow users to track the remaining time for games. For this task, PD11 completion is required. The estimated time for PD12 is around 20 hours.
- **PD13 - AAC system and language setting:** Here, the development of an augmentative and alternative communication (AAC) system will be done, allowing users to communicate within the app. Additionally, language selection settings will be integrated, allowing for voice representation in different languages and offering a wide range of language options for users. The estimated time required for this task is approximately 20 hours and depends on PD4 and PD6.
- **PD14 - Screen adaption**: This part of the development ensures the app adjusts to different screen sizes using appropriate libraries and will be worked on throughout the entire process. The estimated time is about 10 hours. The task can begin once the development environment is fully set up and the UI implementation starts.
- **PD15 - Testing:** Testing and validation of the code will be done in this phase to ensure that widgets and algorithms function correctly without bugs. Testing will

be performed concurrently with the development process. The estimated time for testing is around 50 hours, and it depends on the completion of all previous development tasks.

- **PD16 - Publishing the app on Google Play:** This task involves generating the AAB file in order to publish the app on Google Play, making it available for everyone to download. It depends on PD15 and the use of the Google Play Console [19] for publishing. This task is estimated to take approximately 25 hours.

2.1.3 Project Documentation

- **PDoc1 - Writing the thesis documentation:** The documentation for the final thesis will be written progressively during the project to monitor progress and record key information. The estimated time required for this task is approximately 90 hours.
- **PDoc2 - Preparation of the thesis defense:** This task includes the presentation and the documentation required for the thesis defense. It is estimated to take about 20 hours of work and depends on the completion of PDoc1.

2.2 Resources

In this section, we outline the human and material resources required for this project.

2.2.1 Human Resources

- **Project author:** Is responsible for the entire development process, including the design of the database, mockups, user interface, and more.
- **GEP tutor:** In charge of the initial project management tasks.
- **Thesis supervisor:** Provides feedback and ideas during weekly meetings, ensuring the project progresses correctly and on schedule.

2.2.2 Material Resources

For the development of the project, material resources are required to ensure its successful completion. Below are the necessary resources:

- **Overleaf:** This tool enables us to use LaTeX for formatting and writing high-quality documentation, as well as sharing it easily with our supervisor.

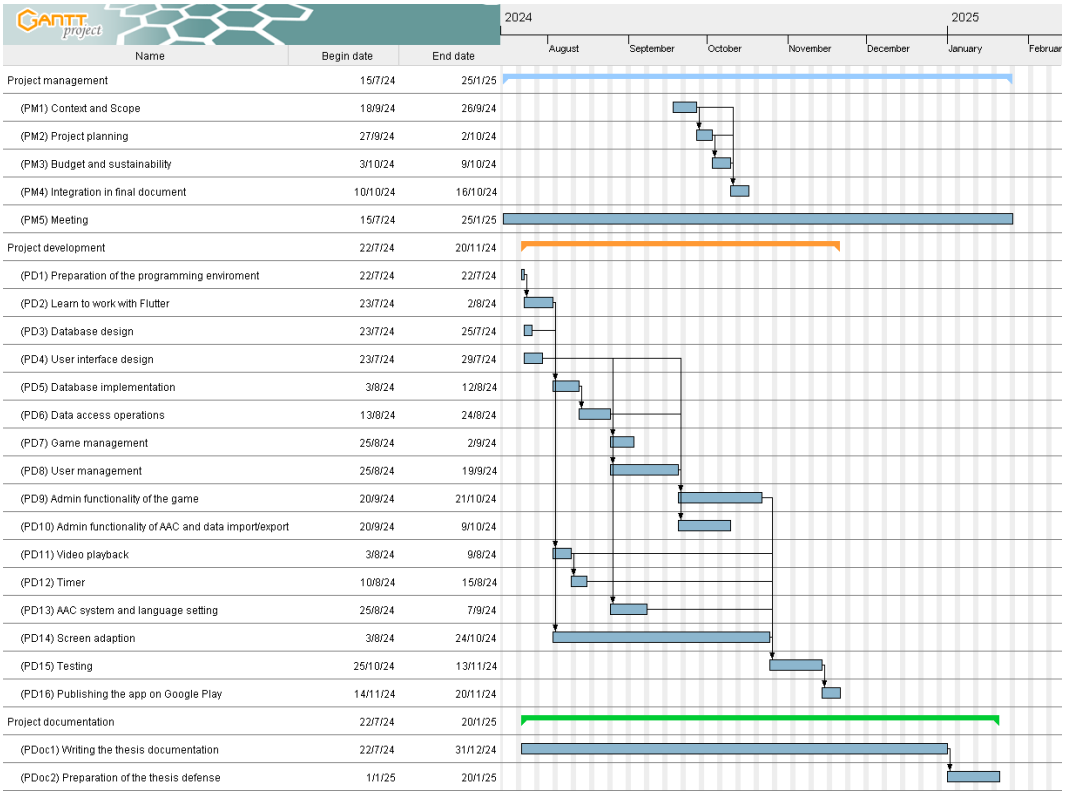
- **GitHub:** A version control tool providing easy access and collaboration features.
- **Figma:** Used for designing the mockups of the user interface.
- **Visual Studio Code:** The integrated development environment (IDE) used for application development offers a wide range of plugins and coding assistance.
- **Android Studio:** This IDE is primarily used for Android application development, but in our project it is mainly used for its Android device emulator.
- **Google Suite:** Tools like Gmail and Google Meet are used for communication and collaboration.
- **Computer and accessories:** This refers to the primary hardware used to develop the entire project, including essential peripherals like the monitor and mouse.
- **GanttProject:** This software is used to create the Gantt chart for project planning.
- **Draw.io:** This tool is used for designing the UML diagram.
- **Google Play Console:** This is a platform that provides a publishing service, allowing us to release our app on Google Play.

2.3 Estimates and Gantt Chart

This section includes an overview table (Table 2.1) estimating the time required for each task, along with a Gantt chart presented in Figure 2.1.

Id	Task	Time estimation(h)	Dependencies	Resources
Project management		**105**	-	
PM1	Context and Scope	30	–	Overleaf, Computer and accessories
PM2	Project planning	20	PM1	Overleaf, Computer and accessories, GanttProject
PM3	Budget and sustainability	20	PM2	Overleaf, Computer and accessories
PM4	Integration in final document	15	PM1, PM2, PM3	Overleaf, Computer and accessories
PM5	Meeting	20	–	Google Meet, Computer and accessories
Project development		**415**	-	
PD1	Preparation of the programming environment	5	–	Android Studio, VS Code, Computer and accessories
PD2	Learn to work with Flutter	25	PD1	Android Studio, VS Code, Computer and accessories
PD3	Database design	5	–	draw.io, Computer and accessories
PD4	User interface design	20	–	Figma, Computer and accessories
PD5	Database implementation	15	PD2, PD3	VS Code, Android Studio, Computer and accessories
PD6	Data access operations	45	PD5	VS Code, Android Studio, Computer and accessories
PD7	Game management	20	PD4, PD6	VS Code, Android Studio, Computer and accessories
PD8	User management	45	PD4, PD6	VS Code, Android Studio, Computer and accessories
PD9	Admin functionality of the game	45	PD4, PD6, PD8	VS Code, Android Studio, Computer and accessories
PD10	Admin functionality of AAC and data export/import	40	PD4, PD6, PD8	VS Code, Android Studio, Computer and accessories
PD11	Video playback	25	PD4	VS Code, Android Studio, Computer and accessories
PD12	Timer	20	PD11	Computer and accessories, VS Code, Android Studio
PD13	AAC system and language setting	20	PD4, PD6	VS Code, Android Studio, Computer and accessories
PD14	Screen adaption	10	PD1 to PD4	VS Code, Android Studio, Computer and accessories
PD15	Testing	50	PD1 to PD14	VS Code, Android Studio, Computer and accessories
PD16	Publishing the app on Google Play	25	PD15	VS Code, Google Play Console, Computer and accessories
Project documentation		**110**	-	
PDoc1	Writing the thesis documentation	90	-	Overleaf, Computer and accessories
PDoc2	Preparation of the thesis defense	20	PDoc1	Overleaf, Computer and accessories
Total		**630**	-	

Table 2.1: Summary of the tasks [Own creation]



Name	Begin date	End date
Project management	15/7/24	25/1/25
(PM1) Context and Scope	18/9/24	26/9/24
(PM2) Project planning	27/9/24	2/10/24
(PM3) Budget and sustainability	3/10/24	9/10/24
(PM4) Integration in final document	10/10/24	16/10/24
(PM5) Meeting	15/7/24	25/1/25
Project development	22/7/24	20/11/24
(PD1) Preparation of the programming enviroment	22/7/24	22/7/24
(PD2) Learn to work with Flutter	23/7/24	2/8/24
(PD3) Database design	23/7/24	25/7/24
(PD4) User interface design	23/7/24	29/7/24
(PD5) Database implementation	3/8/24	12/8/24
(PD6) Data access operations	13/8/24	24/8/24
(PD7) Game management	25/8/24	2/9/24
(PD8) User management	25/8/24	19/9/24
(PD9) Admin functionality of the game	20/9/24	21/10/24
(PD10) Admin functionality of AAC and data import/export	20/9/24	9/10/24
(PD11) Video playback	3/8/24	9/8/24
(PD12) Timer	10/8/24	15/8/24
(PD13) AAC system and language setting	25/8/24	7/9/24
(PD14) Screen adaption	3/8/24	24/10/24
(PD15) Testing	25/10/24	13/11/24
(PD16) Publishing the app on Google Play	14/11/24	20/11/24
Project documentation	22/7/24	20/1/25
(PDoc1) Writing the thesis documentation	22/7/24	31/12/24
(PDoc2) Preparation of the thesis defense	1/1/25	20/1/25

Figure 2.1: Initial Gantt chart [Own creation]

2.4 Risk Management

2.4.1 Deadline of the Project

One of the challenges in time planning is the possibility of underestimating the time required for tasks, which could significantly impact the project. Therefore, it is important to have a proper solution for this situation.

- **Impact:** Medium.
- **Proposed solution:** To address this, I will recalculate the time required for each task. If necessary, I will increase the number of hours dedicated to the project each week to resolve the issue. However, if time remains insufficient, I will prioritize the most critical tasks to ensure the core functionality of the application is completed.

2.4.2 Inexperience with Certain Tools

I have previously used Flutter, but for this project, a new library is required, which may take more time than expected.

- **Impact:** Low.
- **Proposed solution:** The solution is to spend time learning and familiarizing myself with the new library and concepts during the project timeline to reduce any delays.

2.4.3 Bugs

Bugs are a common issue and almost impossible to prevent entirely.

- **Impact:** Medium.
- **Proposed solution:** To manage this, I will continuously perform manual tests to detect bugs as early as possible. This approach will make them easier to identify and fix, speeding up the debugging process. Additionally, I will create automated tests once all the functionality is complete.

2.4.4 Technological Failures

While problems like computer breakdowns are rare, they could significantly delay progress if they occur due to long repair times.

- **Impact:** Medium.
- **Proposed solution:** In case of a breakdown, I will switch to a backup computer. Although setting up the environment on the backup takes some additional time, it will let me continue working without waiting for repairs, which could take a long time. Since all code is stored on GitHub and the documentation in Overleaf, no important data will be lost. The only additional resource required will be the backup computer.

Chapter 3

Budget

In this chapter, we provide a detailed analysis of the total cost and budget for our project. This includes identifying different types of costs, such as human resources, hardware, and software expenses, as well as generic costs like internet, electricity, rent, and more. In addition, to calculate the final cost of the project, we also consider contingencies and any unexpected events.

3.1 Personnel Costs per Activity

In this section, we describe the different roles required to complete the tasks described in the Gantt chart. We also estimate the average salary associated with each role to calculate the total project cost.

In Table 3.1, you can see the required roles along with their corresponding average salaries. Glassdoor [20] provided this information, which is based on the typical annual earnings for these positions.

Role	Annual salary	Gross wage/hour	Gross wage/hour + SS
Flutter developer [FD]	€50.000	€24,04	€31,25
UX/UI Designer [UD]	€30.000	€14,42	€18,75
Project Manager [PM]	€40.000	€19,23	€25,00
QA Tester [QT]	€26.000	€12,50	€16,25

Table 3.1: Estimated costs per role and hour. [Own creation]

In the following table 3.2, we provide more detailed information about the cost of each task.

Id	Task	Time estimation(h)	PM	UD	FD	QT	Cost (€)
Project management		105	-				2625,00
PM1	Context and Scope	30	30	-	-	-	750,00
PM2	Project planning	20	20	-	-	-	500,00
PM3	Budget and sustainability	20	20	-	-	-	500,00
PM4	Integration in final document	15	15	-	-	-	375,00
PM5	Meeting	20	20	-	-	-	500,00
Project development		415	-				11.878,75
PD1	Preparation of the programming environment	5	-	-	4	1	141,25
PD2	Learn to work with Flutter	25	-	-	20	5	706,25
PD3	Database design	5	-	-	5	-	156,25
PD4	User interface design	20	-	20	-	-	375,00
PD5	Database implementation	15	-	-	15	-	468,75
PD6	Data access operations	45	-	-	45	-	1406,25
PD7	Game management	20	-	-	20	-	625,00
PD8	User management	45	-	-	45	-	1406,25
PD9	Admin functionality of the game	45	-	-	45	-	1406,25
PD10	Admin functionality of AAC and data export/import	40	-	-	40	-	1250,00
PD11	Video playback	25	-	-	25	-	781,25
PD12	Timer	20	-	-	20	-	625,00
PD13	AAC system and language setting	20	-	-	20	-	625,00
PD14	Screen adaption	10	-	-	10	-	312,50
PD15	Testing	50	-	-	-	50	812,50
PD16	Publishing the app on Google Play	25	-	-	25	-	781,25
Project documentation		110	-				2.750,00
PDoc1	Writing the thesis documentation	90	90	-	-	-	2.250,00
PDoc2	Preparation of the thesis defense	20	20	-	-	-	500,00
Total		**630**	-				**17.253,75**

Table 3.2: Initial personnel cost for each task defined. [Own creation]

3.2 Generic Cost

These costs are calculated in a general way, as they are not directly related to the specific tasks defined.

3.2.1 Amortization

Hardware

In order to develop our project, some basic hardware components are necessary. The amortization of these material resources is a crucial factor to consider. The amortization cost is calculated using the equation 3.1. In our case, we estimate a useful life of 4 years for these resources, and the duration of the project is six months. A detailed representation of each cost is provided in Table 3.3.

$$\frac{\text{Initial Cost (€)}}{\text{Useful Life (in months)}} \times \text{Duration of the Project (in months)} \tag{3.1}$$

Hardware	Initial cost (€)	Amortization (€)
Asus Rog Strix G15 Laptop	1799,00	224,88
Xiaomi Mi Desktop 1C Display	99,00	12,38
M240 Silent Mouse	29,99	3,75
Total	**€241,01**	

Table 3.3: Amortization of hardware resources [Own creation]

Software

The software used in our project is practically free. However, we need to pay the Google Play Console subscription, which has a one-time fee of $25 USD [21]. For simplicity, we round this amount to **€25**.

3.2.2 Internet Cost

The internet bill costs approximately €15 per month per person. Considering that the project duration is six months with an average of 4.5 hours of work per day, the total cost can be calculated using the following formula:

$$\text{Total Cost} = 6\,\text{months} \times \left(\frac{4.5\,\text{hours}}{24\,\text{hours}}\right) \times 15\,€/\text{month}$$

This results in a total internet cost of approximately **€16,88**.

3.2.3 Electricity Cost

The electricity bill costs €20 per month per person, and the project duration is six months. Given that we work approximately 4.5 hours per day, the total cost can be calculated as follows:

$$\text{Total Cost} = 6\,\text{months} \times \left(\frac{4.5\,\text{hours}}{24\,\text{hours}}\right) \times 20\,€/\text{month}$$

This results in a total electricity cost of approximately **€22,50**.

3.2.4 Work Space

The project is mainly carried out in my apartment located in Barcelona, with a total rent of €900 per month. As the rent is shared with the other two roommates, the cost of the

working space is **€300** per month. Given that the project duration is six months, the total cost of the working space amounts to **€1.800**.

3.2.5 Total Generic Cost

In Table 3.4, the total generic cost (GC) of our project is represented.

Concepts	**Cost (€)**
Amortization	266,01
Internet cost	16,88
Electricity cost	22,50
Work space	1.800,00
Total generic cost (GC)	**2.105,39**

Table 3.4: Total generic cost (GC) [Own creation]

3.3 Other Cost

3.3.1 Contingencies

During the development of the project, unexpected situations may occur, potentially leading to unforeseen costs that could affect our budget. For this reason, it is important to allocate a contingency fund. The contingency will be 15% of the sum of personnel costs per activity (CPA) and generic costs (GC). Therefore, the contingency amount is (€17.253,75 + €2.105,39)*0,15 = **€2.903,87**.

3.3.2 Incidental Costs

Incident	**Estimated cost (€)**	**Risk (%)**	**Cost (€)**
Deadline of the project (30h)	950,00	25	237,50
Inexperience with certain tools (20h)	650,00	15	97,50
Bugs (30h)	950,00	50	475,00
Technological failures	2000,00	5	100,00
Total			**910,00**

Table 3.5: Incidental costs [Own creation]

We also need to consider the additional costs that may arise from implementing alternative plans in case of unexpected events during the project. Therefore, it is essential to account

for these incidental costs in the budget. Table 3.5 presents the total cost to address these obstacles. The cost is calculated by multiplying the probability that each event occurs by its estimated cost.

3.4 Final Budget

The final budget of our project is presented in Table 3.6.

Concepts	Cost (€)
Personnel costs per activity (PCA)	17.253,75
Generic cost (GC)	2.105,39
Contingencies	2.903,87
Incidental cost	910,00
Final budget	**23.173,01**

Table 3.6: Final budget [Own creation]

3.5 Management Control

In large projects, it is difficult to have an accurate estimation of the budget due to the potential occurrence of numerous problems and unexpected events. To manage and control potential budget deviations, we need to define a model to track possible variations in the budget.

Each time a task is completed, we calculate its deviation from the estimated cost and the real cost:

- **Estimated Cost**: The predicted cost of the task done previously.
- **Real Cost**: The actual cost of the task, which requires recalculating the CPA, GC, contingencies, and incidental costs. This allows us to track the real cost and identify whether the task was correctly estimated.
- **Deviation**: The difference between the estimated and real costs of the task.

With the help of this model, we can easily visualize where and why there are deviations, as well as the cost difference. If the deviation is positive, it indicates an overestimation, and the extra budget can be reallocated to another task. With all this information, we can decide whether to utilize part of the contingencies we had prepared.

Chapter 4

Sustainability

4.1 Self-Assessment

At the beginning, I did not expect sustainability to play such a significant role in this final thesis. I always thought it was more relevant to other fields, such as manufacturing, rather than informatics. However, after answering the survey, I realized that there are many aspects of sustainability to consider, including the economic, social, and environmental dimensions, not just the environmental side as I originally thought.

Then I realized that this project could help me understand the footprint my work might generate, along with its social and economic impact. The questions in the following sections also helped us to have a better understanding and analysis of the project's real impact from these three different dimensions.

From an economic perspective, we understand the project's impact through the budget analysis presented in the previous chapter. As for the environmental aspect, although I don't know much about it, I will do my best to minimize its impact. On the social side, this project can truly contribute to society by supporting children with communication difficulties.

4.2 Economic Dimension

Initial Milestone

Regarding PPP: Reflection on the cost you have estimated for the completion of the project

Yes, the cost of the project has been estimated, along with the management and control processes. All of this can be found in the budget chapter.

Regarding Useful Life: How are currently solved economic issues (costs...) related to the problem that you want to address (state of the art)?

Since the project is provided by a professor at UPC, any additional costs would need to be communicated to the project supervisor to negotiate potential financial support. However, most software-related aspects are free. The only potential cost is the Google Play Console service.

How will your solution improve economic issues (costs...) with respect to other existing solutions?

Compared to existing solutions, since this application works offline, it does not require significant WiFi connection costs. Furthermore, this app can reduce the need for physical materials, enable remote therapy sessions, and easily scale to serve many children. These factors can lower long-term costs for educators and therapists, making teaching much more affordable.

Final Milestone

Have you quantified the cost (human and material resources) of carrying out the project? What decisions have you made to reduce the cost? Have you quantified these savings?

We have quantified the cost of human and material resources associated with the project, based on the hours of work and the roles required for each task. The app operates offline, eliminating server maintenance costs. Moreover, by utilizing free software, we further minimized expenses. However, these savings were not qualified, as it is challenging to provide accurate measurements.

Has the estimated cost been adjusted to the final cost? Have you justified the differences (lessons learned)?

Our final cost (€21.737,89) is below the initial planned budget of €23,173.01. However, we used the contingency budget (€2,903.87) from the initial plan to cover the additional personnel costs per activity (PCA). The increase in PCA was primarily due to incorporating feedback received during deployment, which caused some tasks to take longer than expected.

What cost do you estimate the project will have during its useful life? Could this cost be reduced to make it more viable?

If the app is maintained by other developers in the future for bug fixes or additional features, there may be additional costs associated with those tasks. Otherwise, there will be

no additional cost during the app's useful life, as it does not require server maintenance.

Have the costs of adjustments/updates/repairs during the project's useful life been taken into account?

We did not take into account these costs, since it is difficult to predict potential future expenses such as human and material resources. These depend on specific future needs and the involvement of additional developers.

Could scenarios arise that would harm the viability of the project?

There may be scenarios, such as incompatibility with future Android versions, which could require developers to rebuild certain libraries or implement updates.

4.3 Environmental Dimension

Initial Milestone

Regarding PPP: Have you estimated the environmental impact of the project?

It is difficult to estimate its real impact since it will be developed using only one laptop and the environmental impact will be minimal. However, the app itself can significantly reduce the use and need for printed materials or physical tools traditionally used in special education.

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources?

Since the main resource of this app consists of lines of programmed code, there are few resources that can be reused, making it challenging to minimize resource consumption.

How is the problem you want to address currently solved (state of the art). How will your solution be environmentally better than the existing ones?

A lot of existing solutions to help children with communication difficulties rely on physical materials like printed cards or require a constant WiFi connection. This app can reduce the need for printed cards, which require significant materials and need to be reprinted over time. Moreover, the app functions without the need for a constant WiFi connection, making it more efficient and accessible.

Final Milestone

Have you quantified the environmental impact of carrying out the project? What measures have you taken to reduce this impact? Have you quantified this reduction?

We have analyzed the costs of electricity, internet, and related resources, but we did not estimate the exact environmental impact of all the devices used. Since our app operates offline, it reduces the use of internet resources. Additionally, we ensured that all tools and devices used for development were turned off when not in use, but we did not quantify these reductions.

If you were to do the project again, could it be completed with fewer resources?

We could reduce some hardware components, like the monitor and keyboard, using only the laptop and improving initial planning to save some time. However, other resource needs would likely remain unchanged.

What resources do you estimate will be used during the project's useful life? What will be the environmental impact of these resources?

If future developers are involved, they may require all the hardware components for fixing bugs or adding additional features. This could increase energy consumption and internet usage.

Will the project help reduce the use of other resources? Overall, will the project's use improve or worsen the ecological footprint?

The app helps reduce the use of printed materials by replacing them with online resources. Additionally, the lack of a need for a running server minimizes energy consumption, further improving its ecological footprint.

Could scenarios arise that might increase the ecological footprint of the project?

Yes, for example, when the user does not recycle their old device correctly or keeps the app running unnecessarily, it could negatively impact the ecological footprint.

4.4 Social Dimension

Initial Milestone

Regarding PPP: What do you think you will achieve -in terms of personal growth- from doing this project?

I believe this project will help me learn the app development process, understand the different stages, and organize a large-scale project. It will also allow me to better understand the needs and challenges faced by children with communication difficulties, helping me design the app while thinking from their perspective and addressing their needs.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?

Currently, in the market, this problem is solved based on passive learning, where children improve their communication skills by watching specially prepared videos. However, this type of solution lacks interaction.

How will your solution improve the quality of life (social dimension) with respect to other existing solutions?

The current market lacks applications that effectively combine entertainment with the learning process, making learning much more challenging for children. This application can enhance their learning experience and improve their communication skills and finally improve their quality of life and make it easier for them to connect with others.

Regarding Useful Life: Is there a real need for the project?

There is a real need for this project. Currently, there are not many educational apps that combine learning with entertainment. This app can help people with communication difficulties improve their skills, making it easier for them to communicate with others. It will also help them in their daily lives, offering them more opportunities and a better quality of life.

Final Milestone

Has the implementation of this project led to significant reflections on a personal, professional, or ethical level for the individuals involved?

During the project, I learned a lot about the technologies I used to develop the app, as well as how to solve problems and fix bugs more easily and quickly.

Who will benefit from the use of the project? Is there any group that could be negatively affected by the project? To what extent?

Children with communication difficulties will benefit directly from the app, while parents, caregivers, and institutions will benefit indirectly. There are no groups negatively affected by the project, as it is not a commercial app, and its objective is to help children with communication difficulties.

To what extent does the project address the problem initially posed?

The project makes it possible to learn in a fun and engaging way by combining video entertainment and interactive games. It offers a unique experience that makes learning easier and more enjoyable for children. We can confidently say that it solves all the problems initially mentioned and achieves all the objectives.

Could scenarios arise where the project might be harmful to a particular segment of the population?

The app does not create any scenarios that could be harmful to any particular segment of the population. It is designed to be inclusive and suitable for everyone.

Could the project create some form of dependency that leaves users in a vulnerable position?

No, the project is designed to complement existing educational methods, not to replace them. Its goal is to provide additional support.

4.5 Sustainability Matrix

- **Project Deployment (PPP):** This cell is evaluated on a scale from 0 to 10, where 10 represents fully sustainable and 0 means unsustainable.
- **Useful life:** This cell is evaluated on a scale from 0 to 20, where 20 represents fully sustainable throughout its useful life, and 0 means unsustainable.
- **Risks:** This cell is evaluated on a scale from -20 to 0, where 0 represents no identified risks in the three dimensions (social, economic, and environmental), and -20 represents highly dangerous and probable risks.

Table 4.1, represents the sustainability matrix, with a final range between -60 and 90. A score of 90 indicates a fully sustainable project, while -60 represents a completely unsustainable one.

	PPP	Useful Life	Risks
Environmental	Design Consumption (7)	Ecological Footprint (10)	Environmental Risks (-2)
Economic	Invoice (7)	Feasibility Plan (10)	Economic Risks (-2)
Social	Personal Impact (8)	Social Impact (15)	Social Risks (-3)
Sustainability Range	**22**	**35**	**-7**
Total	**50**		

Table 4.1: Sustainability matrix [Own creation].

Chapter 5

Project Monitoring

In this chapter, an analysis of project monitoring is described, including the relevant changes that occurred during project development. This includes scope, working methodology, and updates to planning and budget in relation to initial planning. The objective is to provide a clear overview of the project's progress.

5.1 Scope

For the scope of this project, the objectives we initially defined remained unchanged, and we continued to work based on these established objectives. However, when it comes to the functional requirements, we added a new one called "game difficulty configuration," based on the feedback received during the closed testing phase of the app's release. Additionally, we rewrote some of the functional requirements definitions to make them clearer and easier to understand.

5.2 Methodology

The working methodology we followed during the project remains consistent with the one we established initially, in this case, the Kanban methodology. This approach works well for tracking our progress and represents our current status.

We continued having weekly meetings with the supervisor to coordinate information, seek opinions, and receive feedback from the testers through the supervisor. In addition to the weekly meetings, we maintain email conversations since it is very useful for sharing minor information updates or arranging new meetings.

5.3 Planning

For the time planning of the project, there were some modifications compared to the original schedule since some tasks took longer than expected. As a result, the final time planning had some changes compared to the initial expectations.

The task that took longer than expected was the process of publishing the app on Google Play (task PD16). The reason for this delay was due to the new Google policy, which now requires having a minimum of 20 testers for closed testing. Once the required number of testers was recruited, Google also required the app to undergo testing for 14 days.

All these additional days, combined with the time needed for app review, extended the overall timeline. However, the actual time spent on the publishing process is pretty much the same (25 hours); it was only the overall duration that was extended.

Moreover, to optimize the timeline, we decided to publish the app on Google Play before completing testing. This allows us to collect testers' feedback and make improvements based on their opinions. Based on this feedback, we made some modifications to the app and improved certain features. However, this caused some tasks to take longer than expected, such as **PD7**, **PD12**, **PD13**, and **PD14**, as shown in Figure 5.1.

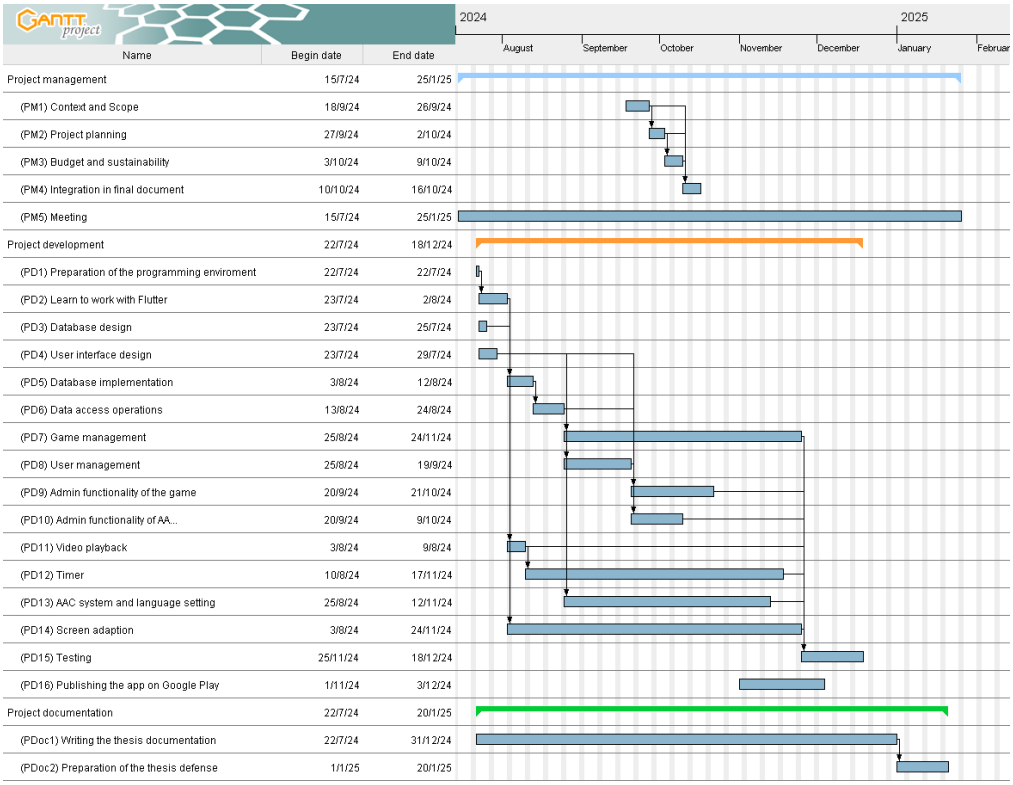


Figure 5.1: Final Gantt chart [Own creation]

Figure 5.1 represents the current Gantt chart, showing all the modifications mentioned previously. We can observe that some tasks took longer than initially planned since the received feedback is related to those tasks. However, there is still enough time to complete the app's features and achieve all the initial plans we have set.

5.4 Budget

Since our initial time plan was modified, the final budget has also been affected by the actual timeline. The following table 5.1 presents the cost associated with each task.

We can observe from Table 5.1 that the final personnel cost for each task amounted to €18.722,50 whereas the estimated personnel cost was €17.253,75. The difference is €18.722,50 - €17.253,75 = €1.468,75, which is well within our contingencies budget of €2.903,87 and this leaves us with a remaining contingency of €1.435,12.

Id	Task	Time estimation(h)	PM	UD	FD	QT	Cost (€)
Project management		105	-				2625,00
PM1	Context and Scope	30	30	-	-	-	750,00
PM2	Project planning	20	20	-	-	-	500,00
PM3	Budget and sustainability	20	20	-	-	-	500,00
PM4	Integration in final document	15	15	-	-	-	375,00
PM5	Meeting	20	20	-	-	-	500,00
Project development		462	-				13.347,50
PD1	Preparation of the programming environment	5	-	-	4	1	141,25
PD2	Learn to work with Flutter	25	-	-	20	5	706,25
PD3	Database design	5	-	-	5	-	156,25
PD4	User interface design	20	-	20	-	-	375,00
PD5	Database implementation	15	-	-	15	-	468,75
PD6	Data access operations	45	-	-	45	-	1406,25
PD7	Game management	40	-	-	40	-	1250,00
PD8	User management	45	-	-	45	-	1406,25
PD9	Admin functionality of the game	45	-	-	45	-	1406,25
PD10	Admin functionality of AAC and data export/import	40	-	-	40	-	1250,00
PD11	Video playback	25	-	-	25	-	781,25
PD12	Timer	30	-	-	30	-	937,50
PD13	AAC system and language setting	35	-	-	35	-	1093,75
PD14	Screen adaption	12	-	-	12	-	375
PD15	Testing	50	-	-	-	50	812,50
PD16	Publishing the app on Google Play	25	-	-	25	-	781,25
Project documentation		110	-				2.750,00
PDoc1	Writing the thesis documentation	90	90	-	-	-	2.250,00
PDoc2	Preparation of the thesis defense	20	20	-	-	-	500,00
Total		**677**	-				**18.722,50**

Table 5.1: Final personnel cost for each task defined. [Own creation]

Table 5.2 shows the actual final budget of the project, with the updated contingency cost, which is calculated as 15% of the CPA and GC. This amounts to (€18.722,50 + €2.105,39)*0.15 = €3.124,19, resulting in a total budget of €24.862,08.

Concepts	Cost (€)
Personnel costs per activity (PCA)	18.722,50
Generic cost (GC)	2.105,39
Contingencies	3.124,19
Incidental cost	910,00
Final budget	**24.862,08**

Table 5.2: Actual final budget [Own creation]

Finally, we calculate the deviations from the initial planning to see the differences in hours and costs compared to our original estimates.

- **Total deviation in hours:** Total hours estimated - Actual hours worked = 630 hours - 677 hours = **-47 hours**.
- **Total deviation in CPA:** Total CPA cost estimated - Actual CPA cost = €17.253,75 - €18.722,50 = **- €1.468,75**.
- **Total deviation in contingencies:** Estimated contingencies - Final contingencies = €2.903,87 - €3.124,19 = **-€220,32**.
- **Total deviation in final budget:** Estimated final budget - Actual final budget = €23.173,01 - €24.862,08 = **-€1.689,07**.

We can observe that all the results are negative, which means that we have underestimated the overall cost.

Chapter 6

App Design

6.1 System Specification

In this section, the use cases for this application are defined based on the functional and non-functional requirements that we have described. The use case provides detailed information about our system and all the functionalities that it offers, helping us to clarify how users will interact with the application.

6.1.1 Actors

As shown in Figure 6.1, there are two types of actors: the admin user and the child user. In this case, the admin user represents the child's parents or caregiver, who is responsible for configuration and other settings to provide a better user experience for the child user.

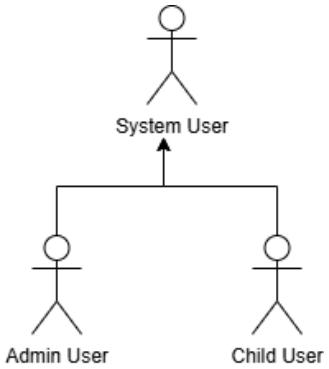


Figure 6.1: Actor hierarchy diagram. [Own creation].

6.1.2 Use Case Diagram

Figure 6.2 shows a diagram that represents all the use cases of our developed app.

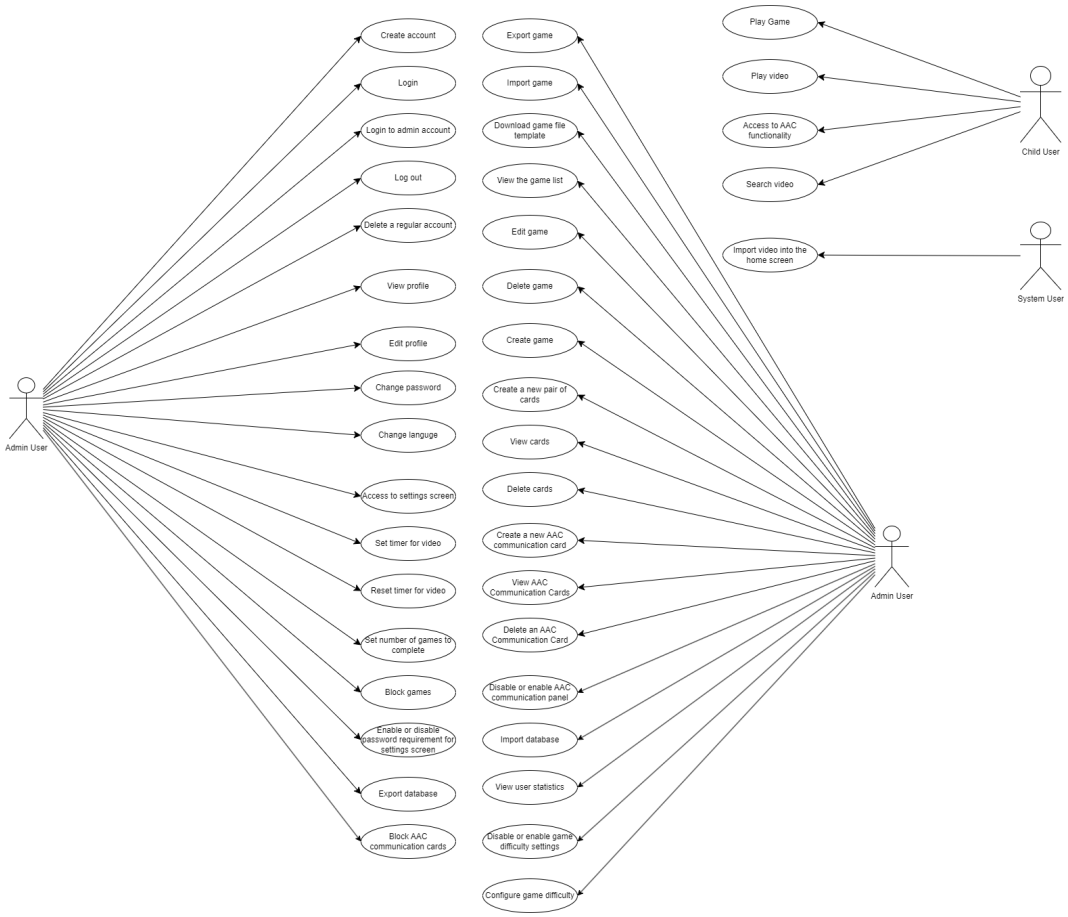


Figure 6.2: Use case diagram. [Own creation].

6.1.3 List of Use Case

- **Create account:** This use case allows users to create their own personal account, allowing them to access the app's functionality later using the corresponding username and password. A full description of this use case can be found in Table B.1.
- **Login:** This allows users to access the app by entering their username and password. If the credentials are correct, the user can have access to all the app's functionalities. If the credentials are incorrect, the app will prompt the user to re-enter the credentials. A more detailed description can be found in Table B.2.

- **Login to admin account:** This use case allows parents or caregivers to access the admin account by entering the corresponding admin username and password. Once logged in, the user can have access to more advanced functionalities, such as CRUD operations for games, creating cards, etc. More information can be found in Table B.3.
- **Change password:** The app allows the user to change their current password. Once the new password is confirmed, the app updates the password in the database. To log in and access the settings screen, a new password will be required. More detailed information can be found in Table B.4.
- **Log out:** The user can log out of the current session and log into another session. Once logged out, the app redirects to the login screen. A detailed description of this use case can be found in Table B.5.
- **Delete a regular account:** This allows the user to delete their account and all associated data from the database. After the deletion, the user will no longer be able to log in with the deleted account. More detailed information can be found in Table B.6.
- **View profile:** Parents or caregivers must be logged in and have access to the settings screen to view the logged user's information, performance, and other important data, such as the number of games played, the number of incorrect guesses, and more. For more information, see Table B.7.
- **Edit profile:** This use case allows users to update their profile information, such as their name, avatar, etc. Once the changes are made, the updated information is saved to the database. For more details, see Table B.8.
- **Change language:** The user can change the language setting of the app and change all content to the selected language to facilitate the app's use. For a more detailed explanation, see Table B.9.
- **Search video:** This use case enables users to easily find the video they are looking for. For more details, see Table B.10.
- **Access to AAC functionality:** The user can access the AAC functionality, which provides support and helps the user to communicate with others by using AAC cards. For more detailed information, see Table B.12.
- **Import video into the home screen:** The user can import videos that are stored locally but are not displaying on the video screen for some reason. For the detailed process, see Table B.13.
- **Access to settings screen:** Parents or caregivers can access the settings screen, where they can modify the app's configurations, such as preferences, the number of games to play, timer settings, and more. For further details, see Table B.14.

- **Set timer for video:** Parents or caregivers can set a timer for the child user, configuring the time for watching the video. Once the timer is finished, the app displays the game. For further information about this configuration, see Table B.15.
- **Clear timer for video:** Parents or caregivers can delete the time limit for watching the video. Once deleted, the child user is free to watch all the videos in the video list without the need to complete the games. For further information about this configuration, see Table B.16.
- **Set number of games to complete:** Parents or caregivers can set the number of games the child user needs to complete when the timer is finished. For more details, see Table B.17.
- **Block games:** Parents or caregivers can choose to block certain games, ensuring they are not shown to the user. This feature can provide a personalized gaming experience to each child user. For more information, see Table B.18.
- **Disable or enable AAC communication panel:** Parents or caregivers can choose to enable or disable the AAC communication panel functionality if it is not necessary. For more information about this configuration, see Table B.19.
- **Block AAC communication cards:** The admin user can choose to block certain communication cards, ensuring that only the unblocked cards are displayed. This helps to create a more personalized and flexible AAC communication panel. For detailed instructions on how this is done, refer to Table B.20.
- **Disable or enable game difficulty settings:** This feature allows parents or caregivers to decide whether the user plays random games (if the setting is disabled) or a more personalized game based on the user's progress and the chosen difficulty level (if enabled). For more information, see Table B.21.
- **Configure game difficulty:** This feature allows parents or caregivers to adjust the game difficulty level to Easy, Medium, or Hard, which is related to the games the user will be playing. Once a difficulty level is selected, the user will only play games that match the chosen difficulty, providing a more personalized experience based on their current progress. For more information see Table B.22.
- **Enable or disable password requirement for settings screen:** Parents or caregivers can decide to enable or disable the password requirement for accessing the settings screen. If enabled, users will be required to enter the current account password to access the settings functionalities. For more information on how this process works, see Table B.23.
- **Import database:** This functionality allows users to import their database from another device to a new one. Since the app is offline, this method allows users to preserve their data even when they change devices. For more details on how it works, see Table B.24.

- **Export database:** Since we offer the functionality to import the database, we must also allow users to export it. This use case enables users to download the current database to their device and later import it to a new device. For further information on how this is done, see Table B.25.
- **Export game:** This use case enables users to export games from the app and save them to their device. Those data can later be imported into other devices. To see how this is processed, see Table B.26.
- **Import Game:** This functionality allows users to import games prepared by others into our device, improving the app's variety of games and enhancing its playability and professionalism. The table B.27 presents more information about how this use case works.
- **Download game file template:** This feature allows users to create games in a more flexible way based on the download template file, although it can only be games without any images and audio. For instructions on how to obtain this template, refer to Table B.28.
- **View the game list:** The app presents a list of the available games, allowing users to view more detailed information about each game and make changes to them. For further information on how to access this feature, refer to Table B.29.
- **Edit game:** The app presents a list of games, allowing users to choose and make modifications to any of these available games. A specific description of this use case is provided in Table B.30.
- **Delete game:** The user can delete the selected game from the game list, provided it is not the only remaining game. Once deleted, the game will be removed from the database and cannot be recovered. For more information on how this is done, check Table B.31.
- **Create game:** In this use case, the user can create a game. The app will provide a form where the user needs to provide all the required information about the game, such as its name, category, and cards. Once verified, the game will be added to the database. For a better understanding of this process, check Table B.32.
- **Create a new pair of cards:** The app provides a form for the user to create a pair of cards for the game, where the user needs to provide all the necessary information and select the card type, either image or text. Once selected, the corresponding fields will appear. After creation, the pair will be added to the database. For more details, check Table B.33.
- **View cards:** The user can view all the available non-panel cards within the app that can be added to the game. Moreover, the app will reflect any updates to the cards that have been created or deleted. For further information, check Table B.34.
- **Delete Cards:** The user can delete non-panel cards from the cards list, and the deleted cards will be permanently removed from the database. For further information on how this is done, check Table B.35.

- **Create a new AAC communication card:** The app provides a form for the user to create an AAC communication card, where the user needs to provide all the necessary information. After creation, the new AAC card will be added to the database, and the user can now use this new card to communicate. For more information, refer to Table B.36.
- **View AAC Communication Cards:** The app allows the user to view all the AAC communication cards, helping them know which ones exist. For more information, see Table B.37.
- **Delete an AAC Communication Card:** The app allows the user to delete selected AAC communication cards, removing the unnecessary ones. Once deleted, the card cannot be recovered. For more information, check Table B.38.
- **Play the game:** The user will be able to play the game prepared after the timer is finished. The user can drag and drop the cards they want to match, helping them learn communication and other skills. For more information, refer to Table B.39.

6.2 UML Design

For our application, it is essential to have a well-designed database to store and manage data effectively. To begin, we implemented a UML class diagram, as shown in Figure 6.3. The diagram consists of four primary classes: **User**, **Game**, **UserGamePlayed**, and **Card**.

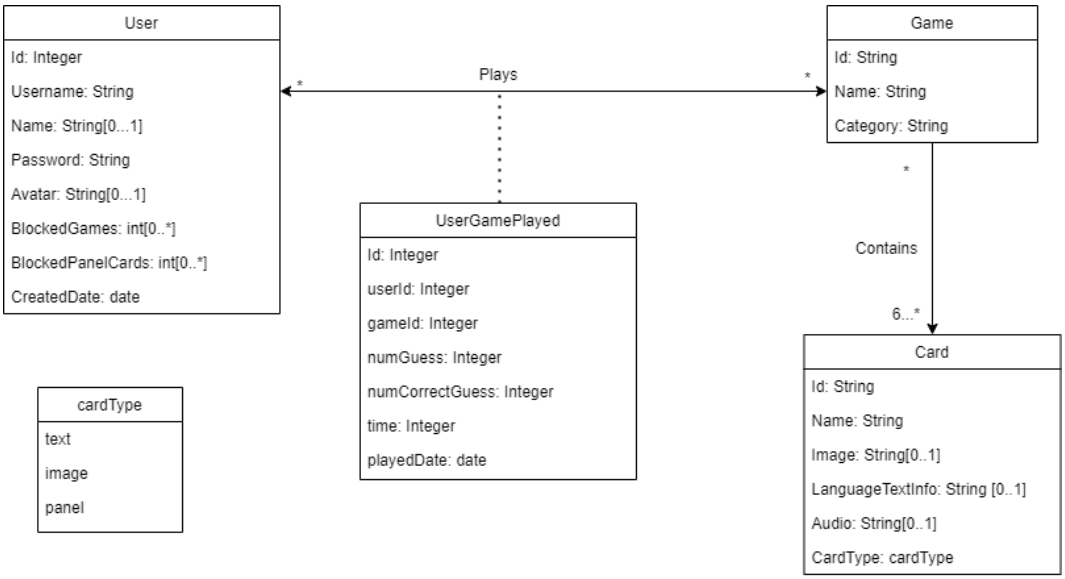


Figure 6.3: UML class diagram [Own creation].

User Class

This class contains all the necessary attributes to store and manage user information, such as username, avatar, and name. It includes an ID attribute, which is the primary key, ensuring that each user has a unique identifier. Additionally, the username attribute is also unique since it is an alternative key, meaning that only one user can have a particular username.

Game Class

This table contains the game information and is composed of the Card class. To create a game, it must include at least 6 non-panel cards, with each card having a matching pair. The ID attribute is the primary key, while the name attribute is the alternative key.

UserGamePlayed Class

The UserGamePlayed class stores the historical data of all the games played by a user and their associated statistics. This includes information such as the number of guesses made, the number of correct guesses, the date the game was played, and the amount of time spent on the game. Although `userId` and `gameId` are foreign keys, the primary key of this table is the ID, allowing multiple records with the same gameId and userId. This design makes it possible to store every game session.

Card Class

The card class contains all relevant information about each card, such as its ID, image, card type, audio, languageTextInfo, and name. The ID attribute serves as the primary key. The class can represent different types: text, image, or panel by using the `cardType` attribute. Panel-type cards are used for AAC and cannot be used for games; both the image and languageTextInfo fields are required. For image-type cards, the image field is required, while the audio is optional. For text-type cards, the languageTextInfo field is required.

Moreover, for the non-panel cards, the name field is used for matching. If two cards share the same name, they are considered matched, so there can only be two non-panel cards with the same name.

In this class, we did not apply the inheritance pattern for each card type because many attributes are shared across different card types. Keeping all attributes in a single table simplifies the database structure, reduces memory usage, improves performance, and makes the data easier to manage.

6.3 App Architecture

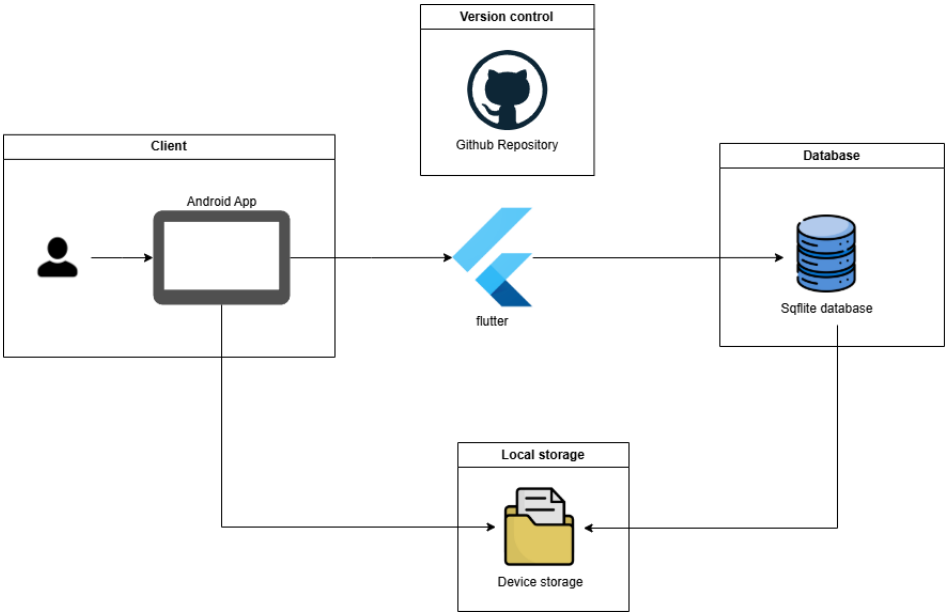


Figure 6.4: App architecture [Own creation].

In Figure 6.4 we can see the architecture of the app, where the user interacts with the application, which is built using **Flutter** to present both the user interface and handling the business logic. Flutter communicates with the locally stored SQLite database to store and retrieve data. Moreover, local storage is used to store certain information, such as images and user preferences, which are not saved in the database.

This structure is designed for an offline app, as all data is stored locally in the SQLite database or local storage. The GitHub repository is used for version control, helping us manage and store the code easily and conveniently.

The architecture used in this application is based on the MVVM (Model-View-ViewModel) pattern. In this case, the **Model** represents the business logic and core data, such as classes that define data, handle data from the SQLite database, and manage CRUD operations. The **View** is responsible for the user interface (UI) and the data presented directly to the user through different widgets and components. The **ViewModel** acts as the connection between the View and the Model; in this case, Flutter's **ChangeNotifier**, **StreamBuilder**, and **Provider** are used to implement the ViewModel, ensuring the correct communication and data flow between Model and View.

This pattern allows for better scalability and facilitates unit testing by separating the responsibilities of the UI, model, and business logic, making the code much easier to maintain. Figure 6.5 provides an example of user interaction within the MVVM architecture.

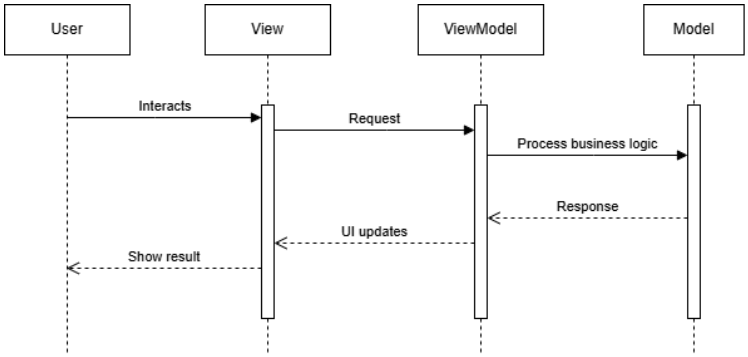


Figure 6.5: User interaction flow in MVVM [Own creation].

6.4 User Interface Mockups

This section presents some of the most important mockups we created to help us have a better understanding and guide us when developing the application. For screens that are primarily used by children, we chose bright and colorful designs to make them more attractive. For functionalities that are used by parents or caregivers, we keep the design simple and clean to make it easier to adjust settings.

6.4.1 Home Screen Mockup

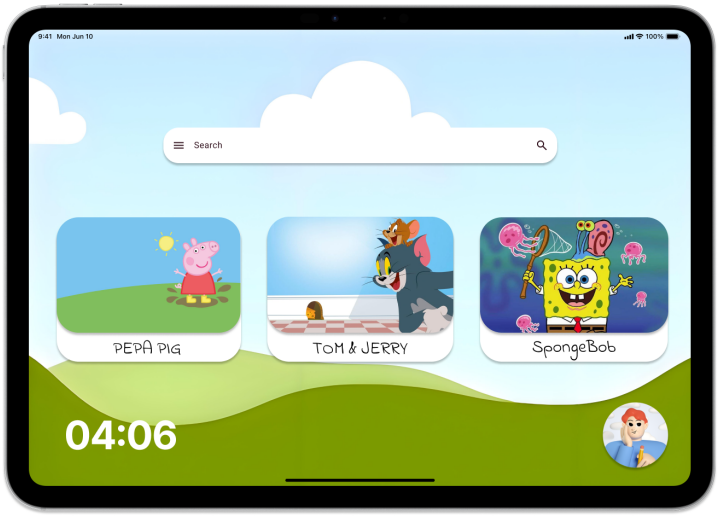


Figure 6.6: Home screen mockup [Own creation].

In Figure 6.6, we have a mockup of the app's home screen, which displays a list of available videos on the device. The user can play a video by clicking on the video card, search for videos using the search bar on the top, and access the settings screen by clicking on the avatar in the bottom right corner. Additionally, a timer is displayed in the bottom-left corner to indicate the remaining time. The background is light and colorful, designed to make the screen more attractive and user-friendly.

6.4.2 Game Screen Mockup

Figure 6.7 represents the game screen, where the user needs to drag and drop the matching cards. In the bottom left corner, there is a timer indicating the time spent on the game, and in the top right corner, the number of games played and the remaining games needed to be completed.

Figure 6.7: Game screen mockup [Own creation].

6.4.3 Login and Register Screens Mockups

Figure 6.8 presents the mockups of the login and registration screens. Since the app is offline, there is no option for third-party login or registration; users can only create an account and log in using the native method. After creating an account with a unique username, the app redirects the user to the login screen. Once logged in, the user is taken to the home screen, as shown in Figure 6.6.

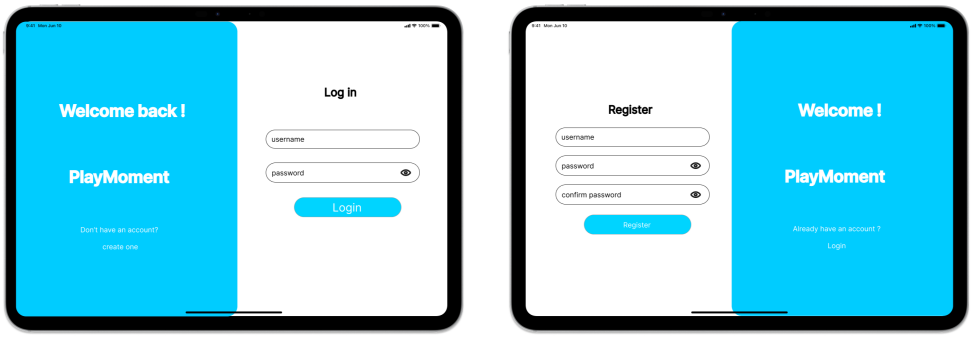


Figure 6.8: Login and register screens mockups [Own creation].

6.4.4 Settings Screen Mockup

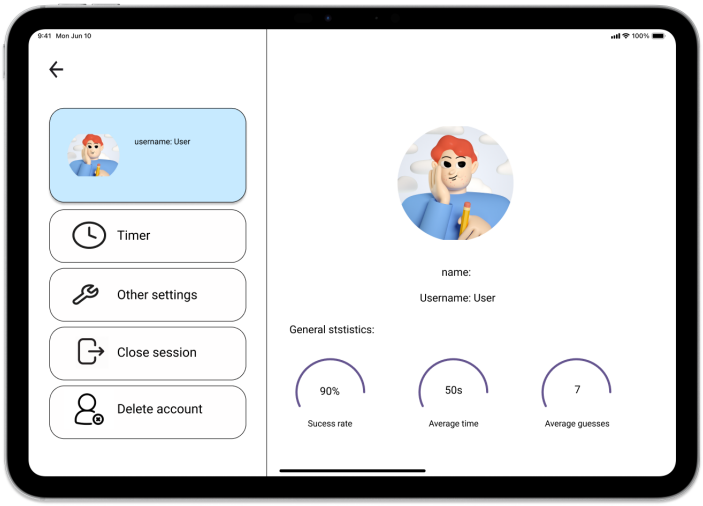


Figure 6.9: Settings screen mockup [Own creation].

Figure 6.9 displays a mockup of the settings screen, where users can configure various settings, such as the timer, profile, logout, delete account, and more, based on the options listed on the left. In this mockup, users can immediately view their data, such as the username, avatar, and statistics like success rate, average time, guesses, etc. By clicking the back arrow in the top left corner, users return to the home screen, as shown in Figure 6.6. The settings screen is designed in a minimalist style, making it easier for users to identify options and functionalities.

6.4.5 Timer Setting Screen Mockup

Figure 6.10 represents the mockup of the timer settings screen, where users can adjust the timer using the slider or the minus and plus icons. Once the preferred time is selected, clicking the "Set timer" button will set the timer, while clicking the "Clear Timer" button will delete it. At the bottom of the timer slider, there are grey and yellow stars. The yellow stars represent the number of games the user wants to play.

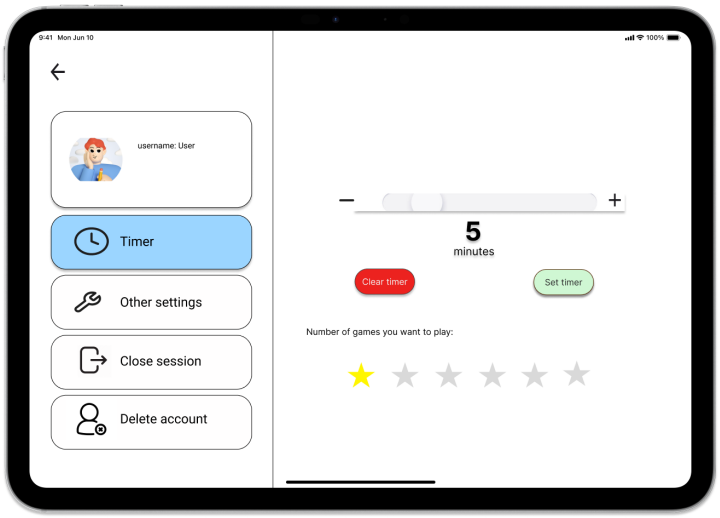


Figure 6.10: Timer setting screen mockup [Own creation].

Chapter 7

App Development

7.1 Technologies

This section defines and explains the technologies used in our app development process.

7.1.1 Flutter

Flutter is an open-source UI software development kit (SDK) created by Google and released in 2017. Today, many big companies, including Alibaba and ByteDance, use this framework for its simplicity and efficiency.

One of Flutter's greatest advantages is that it allows developers to build natively compiled applications for mobile (both iOS and Android), web, and desktop from a single codebase. This means developers only need to write code once, and the app can be compiled and run on multiple platforms without needing platform-specific adjustments. This feature simplifies the development process, saves time, and reduces unnecessary costs.

In our project, we used Flutter to develop our application. Although Flutter supports creating apps for multiple platforms, we focused only on the Android version. Since our app is offline, we used Flutter not only for the frontend but also to handle the backend functions of the app.

The reason for choosing this technology to develop the app is that we have previously worked with it, so it saves some time in the learning process. Moreover, it offers a hot reload feature that allows us to see changes in real time that facilitated the UI development. Flutter also provides various native widgets and has a large support community that makes it easier to solve issues. Moreover, Flutter apps can perform almost like native apps, offering a smooth user experience with minimal lag.

Key Features

- **Widget-based Framework:** The entire Flutter UI is widget-based, and it supports customizable widgets that enable developers to create flexible, beautiful, and user-friendly designs.
- **Hot Reload:** This feature lets developers instantly see changes that they have made, making it easier to detect bugs and speeding up the development process.
- **Dart Language:** Flutter uses Dart as its programming language, which is optimized for UI development and has fast compilation times.
- **High Performance:** Flutter apps run directly on the device's hardware, resulting in smooth animations and excellent performance.

7.1.2 Dart Language

Dart is the programming language used by the Flutter framework, and it was deeply influenced by languages such as C, C++, and Java. Created by Lars Bak and Kasper Lund in 2011, developed by Google, and officially released in 2013.

This programming language is known for its class-based structure, support for hot reload, which allows developers to view the changes instantly without restarting the app, and built-in support for asynchronous programming by using async and await. Its clean syntax makes it easy for developers to learn and create complex UIs efficiently. Moreover, Dart offers AOT compilation at build time, which reduces runtime processing and enables cross-platform compatibility so it can run on various platforms such as iOS, Android, and more.

7.2 Code Structure

In this section, we present the code structure of our application. The project is organized into the following directories for better organization.

- **lib/**: Contains all the core application code, such as database management, models, utilities, and user interface components.
- **assets/**: Contains all other important files, such as images, initial game data, fonts, music, and more.
- **test/**: Contains unit and widget tests of the code to ensure the correct functionality of the app.

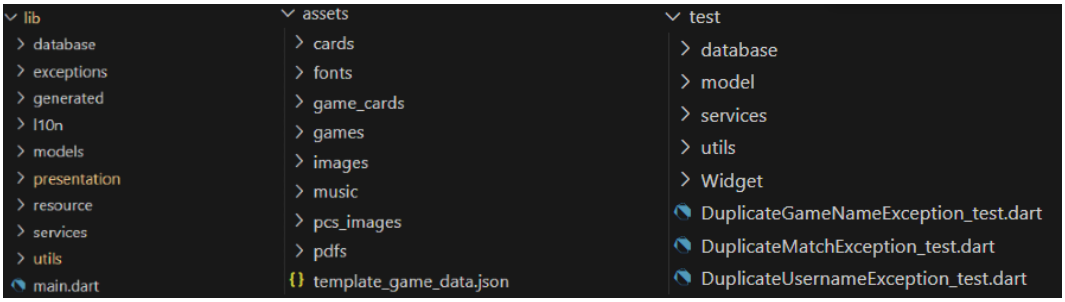


Figure 7.1: Code structure directories [Own creation].

In Figure 7.1, there is a detailed presentation of the **lib**, **assets**, and **test** directories.

The **lib/** directory is organized into the following folders:

- **Database folder:** This folder contains all the necessary tables for the app, such as User, Game, Card, etc. It also includes data access operations that allow us to perform CRUD operations on these tables.
- **Generated and l10n folders:** The **generated** folder contains all the auto-generated code based on the translations and the dictionary from the **l10n** folder. The **l10n** folder contains all the translations for the languages we have set, making possible the localization of our app.
- **Models folder**: This folder contains the models used to represent the tables in the database folder.
- **Exceptions folder:** This folder contains the different types of exceptions that we created to improve the exception management of our app.
- **Presentation folder:** This folder contains all the UI we have created to interact with the user, including widgets and screens.
- **Resource folder:** This folder contains the color and font size resources used in our application, ensuring a standardized look and feel for the app.
- **Services:** This folder contains all the services we created related to **ChangeNotifier**, **StreamController**, and **SharedPreferences**, which help connect the business logic with the user interface.
- **Utils:** This folder has all the functionalities and algorithms that we designed to handle the logic, including tasks like accessing and modifying data.

The **assets/** directory is organized into the following folders:

- **Cards:** This folder contains all the initial card data that will be inserted into the database.
- **Games:** This folder contains the initial game data that will be inserted into the database.
- **Game cards:** This folder contains the relational information between cards and games, which will also be inserted into the database.
- **Fonts:** This folder contains the font files used in our app.
- **Images:** This folder contains all the images that are not used for the initial games, such as the default avatar, app logo, background, and more.
- **Music:** This folder contains the music resources used in the app.
- **Pcs_images:** This folder contains all the images used in the initial games.
- **Pdfs:** This folder contains the user manual in PDF format, which is available within the app.

The **test/** directory contains folders such as database, model, services, utils, and widgets, which test the contents of the corresponding folders in the **lib** directory.

7.3 Version Control

For version control, our application follows a branching strategy where a main feature branch, named **develop**, is created from the **main** branch. From the **develop** branch, we develop different functionalities, features, and bug fixes. Once all functionalities are completed and tested, the final **develop** branch is merged into **main**, as shown in Figure 7.2.

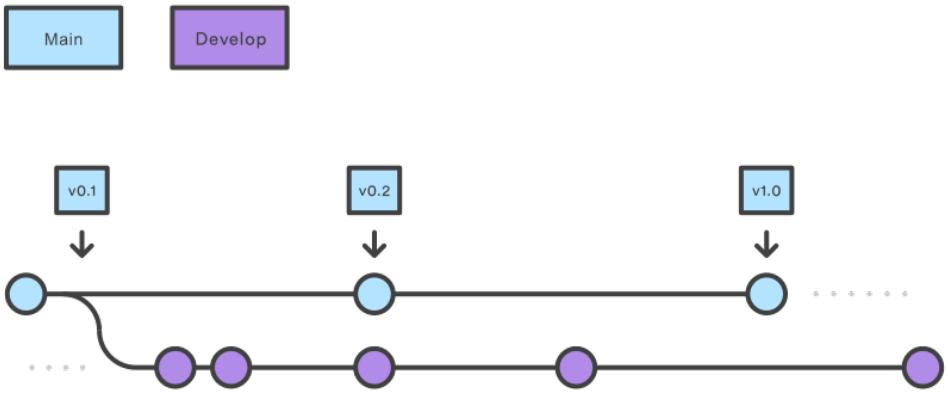


Figure 7.2: Git workflow [22].

We adopted this strategy because many features were closely related, and having them in separate branches made the development process more complicated. Additionally, since the project is being developed by just one developer, it is more reasonable to work in a single branch; having multiple branches wouldn't really improve performance, since it is impossible to work on different features at the same time.

7.4 Database Tables

To implement the UML diagram we designed, we use the **sqflite** library provided by Flutter, which allows us to create an SQLite database. Since our project is relatively small, SQLite is sufficient and provides all the necessary functions and tools to create the required database and tables.

The database contains five tables: **GameTable**, **UserTable**, **CardTable**, **GameCardTable**, and **UserGamePlayedTable**. SQLite supports only basic data types, such as NULL, INTEGER, TEXT, REAL, and BLOB. Consequently, attributes of the List type in our UML diagram (Figure 6.3) were changed to the TEXT type.

- **User Table:** This table stores all the user's personal information, such as username, name, password, avatar, etc. The password attribute is securely stored using cryptographic hashing, and the avatar attribute stores the path of the image on the device instead of the image itself to minimize storage usage. The primary key of this table is `id`, and the `username` attribute is also unique.
- **Game Table:** This table contains information about the games, including the game name, category, and `id`. The primary key of this table is `id`.
- **UserGamePlayed Table:** This table stores the game history for each user. It has the `game_id` and `user_id` as foreign keys, while the primary key is `id`. This design allows the database to store multiple instances of the same user playing the same game, having each record stored separately.
- **Card Table:** This table stores information about the cards, which can be panel cards, text cards, or image cards. Each card has its corresponding attributes, with relevant fields containing non-null values and other irrelevant fields set to null.
- **GameCard Table:** This table represents the relationship between games and cards, indicating which cards are associated with the corresponding game. So the `game_id` and `card_id` are the primary key.

The following table provides an overview of all attributes in the tables (Figure 7.3). All these tables are stored locally on the user's device to ensure the app can function offline.

```dart
class GameTable {
  static const String GAME_TABLE_NAME = 'game';
  static const String GAME_COLUMN_ID = 'id';
  static const String GAME_COLUMN_NAME = 'name';
  static const String GAME_COLUMN_CATEGORY = 'category';

  static Future<void> createTable(Database database, int version) async {
    await database.execute('''
      CREATE TABLE $GAME_TABLE_NAME (
        $GAME_COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
        $GAME_COLUMN_NAME TEXT NOT NULL UNIQUE,
        $GAME_COLUMN_CATEGORY TEXT NOT NULL
      )
    ''');
  }
}
```

```dart
class GameCardTable {
  static const String GAME_CARD_TABLE_NAME = 'game_card';
  static const String GAME_CARD_COLUMN_GAME_ID = 'game_id';
  static const String GAME_CARD_COLUMN_CARD_ID = 'card_id';

  static Future<void> createTable(Database database, int version) async {
    await database.execute('''
      CREATE TABLE $GAME_CARD_TABLE_NAME (
        $GAME_CARD_COLUMN_GAME_ID INTEGER NOT NULL,
        $GAME_CARD_COLUMN_CARD_ID INTEGER NOT NULL,
        FOREIGN KEY ($GAME_CARD_COLUMN_GAME_ID) REFERENCES game(id) ON DELETE CASCADE,
        FOREIGN KEY ($GAME_CARD_COLUMN_CARD_ID) REFERENCES card(id) ON DELETE CASCADE,
        PRIMARY KEY ($GAME_CARD_COLUMN_GAME_ID, $GAME_CARD_COLUMN_CARD_ID)
      )
    ''');
  }
}
```

```dart
class UserTable {
  static const String USER_TABLE_NAME = 'user';
  static const String USER_COLUMN_ID = 'id';
  static const String USER_COLUMN_USERNAME = 'username';
  static const String USER_COLUMN_NAME = 'name';
  static const String USER_COLUMN_PASSWORD = 'password';
  static const String USER_COLUMN_AVATAR = 'avatar';
  static const String USER_CREATED_TIME = 'created_time';
  static const String USER_COLUMN_BLOCK_GAMES = 'block_games';
  static const String USER_COLUMN_BLOCK_PANEL_CARDS = 'block_panel_cards';

  static Future<void> createTable(Database database, int version) async {
    await database.execute('''
      CREATE TABLE $USER_TABLE_NAME (
        $USER_COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
        $USER_COLUMN_USERNAME TEXT NOT NULL UNIQUE,
        $USER_COLUMN_NAME TEXT,
        $USER_COLUMN_PASSWORD TEXT NOT NULL,
        $USER_COLUMN_AVATAR TEXT,
        $USER_COLUMN_BLOCK_GAMES TEXT,
        $USER_COLUMN_BLOCK_PANEL_CARDS TEXT,
        $USER_CREATED_TIME TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP
      )
    ''');
  }
}
```

```dart
class CardTable {
  static const String CARD_TABLE_NAME = 'card';
  static const String CARD_COLUMN_ID = 'id';
  static const String CARD_COLUMN_NAME = 'name';
  static const String CARD_COLUMN_IMAGE = 'image';
  static const String CARD_COLUMN_CARD_TYPE = 'cardType';
  static const String CARD_COLUMN_LANGUAGUE_INFO = 'languageInfo';
  static const String CARD_COLUMN_AUDIO = 'audio';

  static Future<void> createTable(Database database, int version) async {
    await database.execute('''
      CREATE TABLE $CARD_TABLE_NAME (
        $CARD_COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
        $CARD_COLUMN_NAME TEXT NOT NULL,
        $CARD_COLUMN_IMAGE TEXT,
        $CARD_COLUMN_CARD_TYPE TEXT NOT NULL,
        $CARD_COLUMN_LANGUAGUE_INFO TEXT,
        $CARD_COLUMN_AUDIO TEXT
      )
    ''');
  }
}
```

```dart
class UserGamePlayedTable {
  static const String USER_GAME_PLAYED_TABLE_NAME = 'user_game_played';
  static const String USER_GAME_PLAYED_COLUMN_ID = 'id';
  static const String USER_GAME_PLAYED_COLUMN_USER_ID = 'user_id';
  static const String USER_GAME_PLAYED_COLUMN_GAME_ID = 'game_id';
  static const String USER_GAME_PLAYED_COLUMN_NUM_GUESS = 'num_guess';
  static const String USER_GAME_PLAYED_COLUMN_NUM_CORRECT_GUESS = 'num_correct_guess';
  static const String USER_GAME_PLAYED_COLUMN_TIME = 'time';
  static const String USER_GAME_PLAYED_COLUMN_DATE = 'played_date';

  static Future<void> createTable(Database database, int version) async {
    await database.execute('''
      CREATE TABLE $USER_GAME_PLAYED_TABLE_NAME (
        $USER_GAME_PLAYED_COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
        $USER_GAME_PLAYED_COLUMN_USER_ID INTEGER NOT NULL,
        $USER_GAME_PLAYED_COLUMN_GAME_ID INTEGER NOT NULL,
        $USER_GAME_PLAYED_COLUMN_NUM_GUESS INTEGER NOT NULL DEFAULT 0,
        $USER_GAME_PLAYED_COLUMN_NUM_CORRECT_GUESS INTEGER NOT NULL DEFAULT 0,
        $USER_GAME_PLAYED_COLUMN_TIME INTEGER NOT NULL DEFAULT 0,
        $USER_GAME_PLAYED_COLUMN_DATE TEXT NOT NULL,
        FOREIGN KEY ($USER_GAME_PLAYED_COLUMN_USER_ID) REFERENCES ${UserTable.USER_TABLE_NAME}(${UserTable.USER_COLUMN_ID}) ON DELETE CASCADE,
        FOREIGN KEY ($USER_GAME_PLAYED_COLUMN_GAME_ID) REFERENCES ${GameTable.GAME_TABLE_NAME}(${GameTable.GAME_COLUMN_ID}) ON DELETE CASCADE
      )
    ''');
  }
}
```

Figure 7.3: Database tables [Own creation].

7.5 Game Difficulty Algorithm

Before explaining the game difficulty algorithm, it is important to know that there are three difficulty levels: **Easy**, **Medium**, and **Difficult**. Each game is assigned to a corresponding level based on the user's accuracy in the game. For games that the user has

never played before, the accuracy is set to 50%, which corresponds to the medium difficulty.

The accuracy of the game is calculated using Equation 7.1, and the difficulty level is assigned as follows:

- **Easy:** Accuracy is >= 70%.
- **Medium:** Accuracy is between >= 50% and < 70%.
- **Difficulty:** Accuracy is < 50%.

$$\text{Accuracy} = \frac{\text{Number of Correct Guesses}}{\text{Total Number of Guesses}} \times 100 \tag{7.1}$$

In our algorithm, the app randomly selects the games that match the current difficulty level. If more than one game needs to be played, the current game will not be the same as the previous one, except when only one game is available.

The default game difficulty is set to **Medium**, meaning that initially, all games are available to users who have not played before. As the user continues to play, the algorithm will gradually improve, since the game's accuracy is based on the user's performance.

If no games are available for the selected game difficulty level, the app will select games from other difficulty levels in a predefined order. For example:

- For **Easy**, the order is: **Easy** → **Medium** → **Hard**.
- For **Medium**, the order is: **Medium** → **Easy** → **Hard**.
- For **Hard**, the order is: **Hard** → **Medium** → **Easy**.

Since our app does not allow users to delete all the games, there will always be at least one game available to play. This can ensure the user experience and the app's main functionality.

Chapter 8

Testing

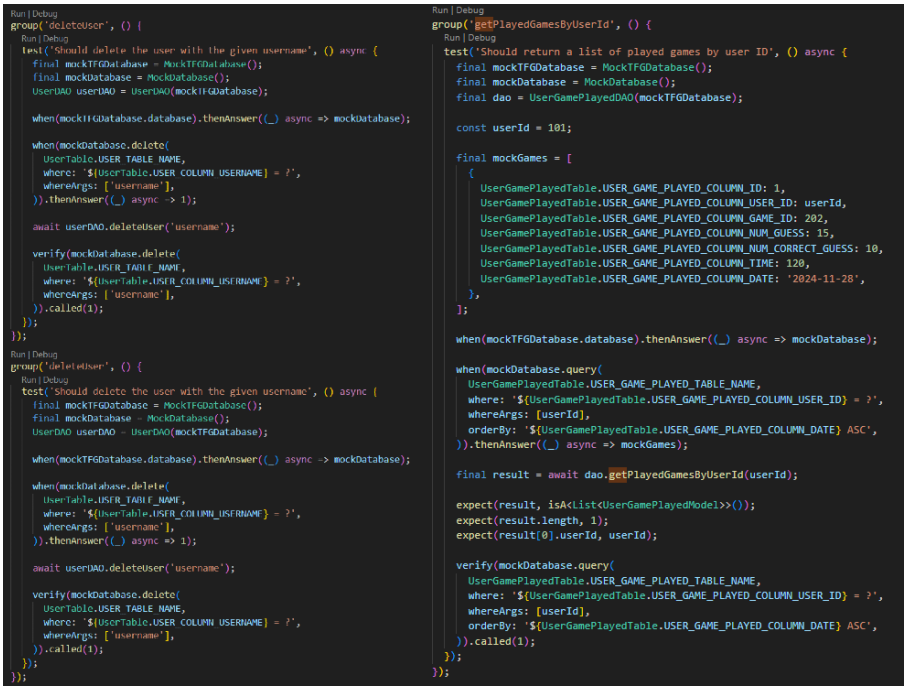
This chapter describes the testing methodology used to test the app. There are various types of tests, such as unit tests, integration tests, etc. However, for our application, the focus is mainly on unit testing, widget testing, and manual testing.

8.1 Unit Testing

This type of testing consists of testing individual parts or components of the app. It is typically applied to isolated functions to ensure that each part works correctly in an isolated environment. In our project, we applied this testing to the files located in the database, model, and services folders within the test directory (see Figure 7.1).

To perform unit testing on the app's functions, we used libraries provided by Flutter, such as flutter_test for function testing and mockito for generating mocks. Our unit tests primarily focused on data access operations related to the database, since they are crucial to ensuring that data is stored, updated, and retrieved correctly. For example, we tested functions like user creation, game creation, card modification, and card deletion to verify their expected behavior (Figure 8.1).

Achieving 100% test coverage is ideal but often challenging due to code complexity and time constraints. For this project, our goal is to achieve at least 70% of test coverage to ensure that most critical functions are tested. Figure 8.2 shows the test coverage achieved for each part of the project.



```dart
Run | Debug
group('deleteUser', () {
  Run | Debug
  test('Should delete the user with the given username', () async {
    final mockTFGDatabase = MockTFGDatabase();
    final mockDatabase = MockDatabase();
    UserDAO userDAO = UserDAO(mockTFGDatabase);

    when(mockTFGDatabase.database).thenAnswer((_) async => mockDatabase);

    when(mockDatabase.delete(
      UserTable.USER_TABLE_NAME,
      where: '${UserTable.USER_COLUMN_USERNAME} = ?',
      whereArgs: ['username'],
    )).thenAnswer((_) async => 1);

    await userDAO.deleteUser('username');

    verify(mockDatabase.delete(
      UserTable.USER_TABLE_NAME,
      where: '${UserTable.USER_COLUMN_USERNAME} = ?',
      whereArgs: ['username'],
    )).called(1);
  });
});

Run | Debug
group('deleteUser', () {
  Run | Debug
  test('Should delete the user with the given username', () async {
    final mockTFGDatabase = MockTFGDatabase();
    final mockDatabase = MockDatabase();
    UserDAO userDAO = UserDAO(mockTFGDatabase);

    when(mockTFGDatabase.database).thenAnswer((_) async => mockDatabase);

    when(mockDatabase.delete(
      UserTable.USER_TABLE_NAME,
      where: '${UserTable.USER_COLUMN_USERNAME} = ?',
      whereArgs: ['username'],
    )).thenAnswer((_) async => 1);

    await userDAO.deleteUser('username');

    verify(mockDatabase.delete(
      UserTable.USER_TABLE_NAME,
      where: '${UserTable.USER_COLUMN_USERNAME} = ?',
      whereArgs: ['username'],
    )).called(1);
  });
});
```

```dart
Run | Debug
group('getPlayedGamesByUserId', () {
  Run | Debug
  test('Should return a list of played games by user ID', () async {
    final mockTFGDatabase = MockTFGDatabase();
    final mockDatabase = MockDatabase();
    final dao = UserGamePlayedDAO(mockTFGDatabase);

    const userId = 101;

    final mockGames = [
      {
        UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_ID: 1,
        UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_USER_ID: userId,
        UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_GAME_ID: 202,
        UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_NUM_GUESS: 15,
        UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_NUM_CORRECT_GUESS: 10,
        UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_TIME: 120,
        UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_DATE: '2024-11-28',
      },
    ];

    when(mockTFGDatabase.database).thenAnswer((_) async => mockDatabase);

    when(mockDatabase.query(
      UserGamePlayedTable.USER_GAME_PLAYED_TABLE_NAME,
      where: '${UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_USER_ID} = ?',
      whereArgs: [userId],
      orderBy: '${UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_DATE} ASC',
    )).thenAnswer((_) async => mockGames);

    final result = await dao.getPlayedGamesByUserId(userId);

    expect(result, isA<List<UserGamePlayedModel>>());
    expect(result.length, 1);
    expect(result[0].userId, userId);

    verify(mockDatabase.query(
      UserGamePlayedTable.USER_GAME_PLAYED_TABLE_NAME,
      where: '${UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_USER_ID} = ?',
      whereArgs: [userId],
      orderBy: '${UserGamePlayedTable.USER_GAME_PLAYED_COLUMN_DATE} ASC',
    )).called(1);
  });
});
```

Figure 8.1: Unit testing examples [Own creation].

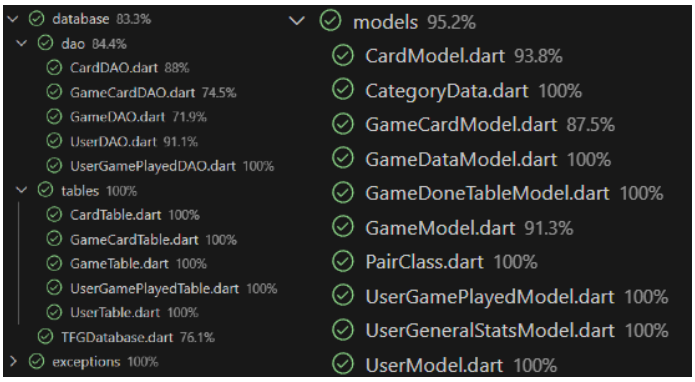


Figure 8.2: Test coverage [Own creation].

8.2 Widget Testing

This type of testing focuses on testing the simple widgets we have created, such as buttons, text fields, loading assets, dialogs, etc. These are general components used throughout the app. This testing helps us analyze whether they are displayed correctly and function as expected. To make this type of test, we used the flutter_test library.

8.3 Manual Testing

This type of testing consists of testing the application manually; this allows us to better identify bugs or problems based on real usage scenarios. Sometimes, it is challenging to make an automated test for the widgets because they are composed of multiple interconnected widget components, making automated testing less practical. So by doing manual testing, we can simulate real user interactions, which makes it easier to identify and fix bugs. In the appendix C, there is a general description of the manual tests we have conducted on our application.

Chapter 9

App Deployment

This chapter describes the deployment process for publishing the app on Google Play, making it accessible to billions of users. The deployment process is managed through the Google Play Console and consists of two stages: **closed testing** and **app publishing**.

9.1 Closed Testing

To publish the app on Google Play, the first step is to pass the closed testing phase. This stage consists of recruiting 20 testers to use the app over a 14-day period. The main objective is to allow users to test the app and provide feedback. Based on this feedback, necessary modifications can be made to ensure the app meets quality standards before its final release.

To recruit these 20 testers, we asked our friends and family. Moreover, we promoted the app testing in several school WhatsApp groups to meet the required number of testers, since 20 participants are needed for this process.

For us, the closed testing is a great opportunity because it allows us to analyze whether the testers are using all the features our app offers and to observe if they are interacting with the app as we expected. Additionally, it provides valuable feedback and suggestions from the testers, helping us to improve the app before the official release.

During the closed testing period, we observed that the testers were using the app as expected. We received feedback through the email linked to the app as well as from the WhatsApp groups where we promoted the app. The feedback mainly focused on the app's ease of usage and feature suggestions.

Based on the feedback received, we improved certain features that were unclear to provide

a better user experience. Additionally, we expanded the educational content and updated the database of educational tasks to better meet users' needs. Once all feedback was collected and addressed, we determined that the app was ready for production. Moreover, the testers confirmed that the app functions as intended and provides a meaningful learning experience for the target audience, indicating that it meets the standard for release.

9.2 App Publishing

Based on the feedback received, we confirmed that the app functions as expected and successfully completed the 14-day closed testing period. Following this, we transitioned the app to production. The app is now available on Google Play under the name **"PlayMoment"**.

Additionally, the Google Play Console (Figure 9.1) allows us to monitor user activity and track metrics such as installations, updates, and user demographics by country. This helps us to analyze the app's performance more effectively.

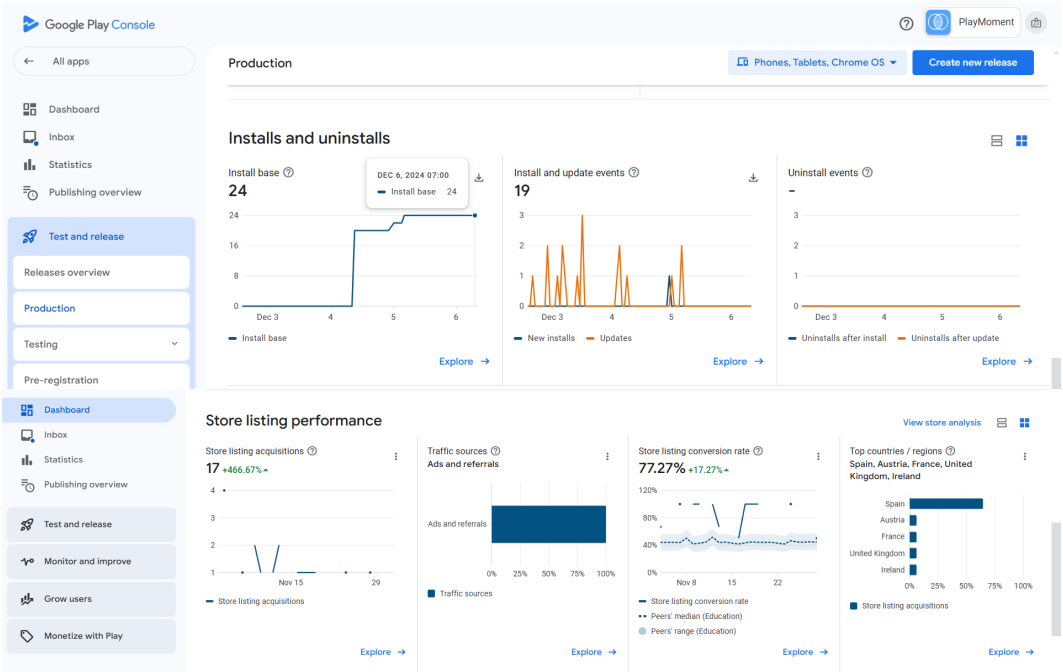


Figure 9.1: Google Play Console [Own creation].

Chapter 10

App Screens and User Guide

In this chapter, the final app's screens are displayed along with a detailed explanation of how to navigate and use the provided functionality, serving as a user guide.

To download the application, go to Google Play, search for "PlayMoment", and install it. Once installed, users can access all the app's features (Figure 10.1).

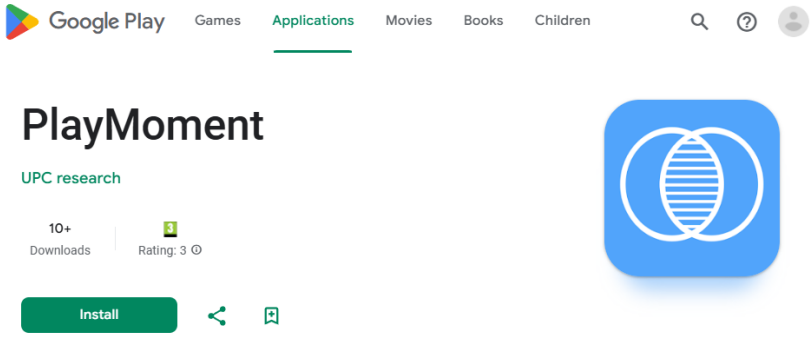


Figure 10.1: PlayMoment on Google Play [Own creation].

10.1 Login and Register Screens

The register and login screens are used to create and authenticate a user. To authenticate, we first need to create an account, and once the account is created, the app navigates to the login screen, where we can log in using the newly created account. Figure 10.2 presents both the Login and Register screens.

- Clicking the button labeled 1 navigates to the login screen.
- Clicking the button labeled 2 navigates to the register screen.
- Clicking the world icon in the top right corner of the login screen displays a list of available languages for the app. Once a language is selected, the app content updates automatically. By default, the app is set to English.
- Clicking the database icon in the bottom right corner of the login screen displays a confirmation dialog for importing the database. Once confirmed, the app opens the file manager, where the user needs to select the database file to be imported.

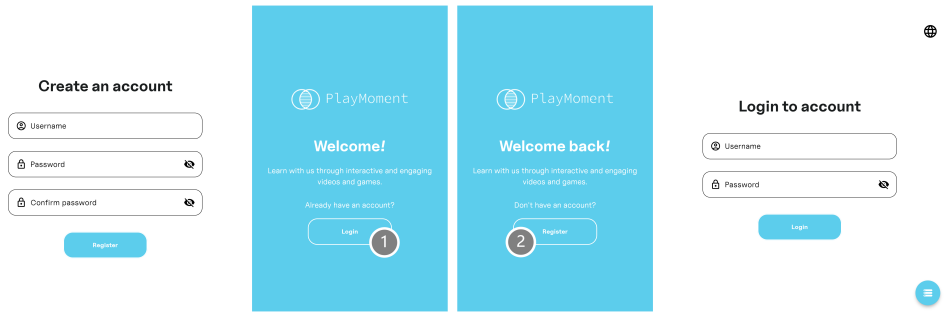


Figure 10.2: Login and register screens [Own creation].

10.2 Main Screen

Figure 10.3 represents the app's home screen with the following numbered elements:

1. **Search Bar:** Users can search for videos by clicking the search bar.
2. **Import video icon:** Users can import the videos using this icon.
3. **AAC Communication panel option:** Clicking this icon navigates to the AAC communication panel screen (Figure 10.4).
4. **Avatar Image:** Clicking this image navigates to the settings screen, but it requires password authentication (Figure 10.4).
5. **Video Cards:** Users can play the displayed videos by selecting the corresponding video card.
6. **Remain Timer:** Indicates the remaining timer.

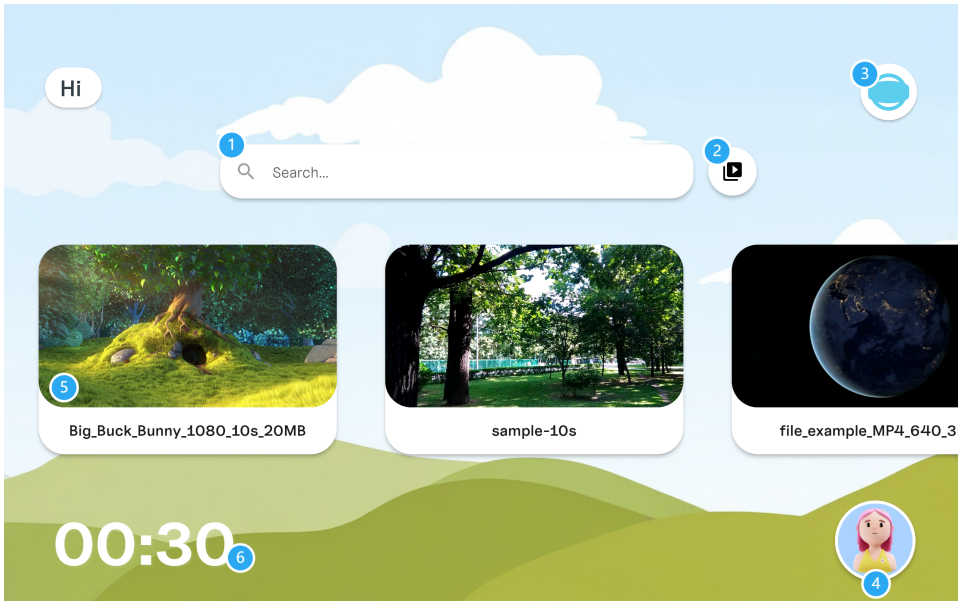


Figure 10.3: Home screen [Own creation].

Figure 10.4 shows the AAC communication panel screen and the password authentication required to access the settings screen, preventing children from accessing the settings options. In the AAC communication panel screen, daily basic vocabularies are presented where the user can press a select card to produce the corresponding audio based on the app's current language setting.

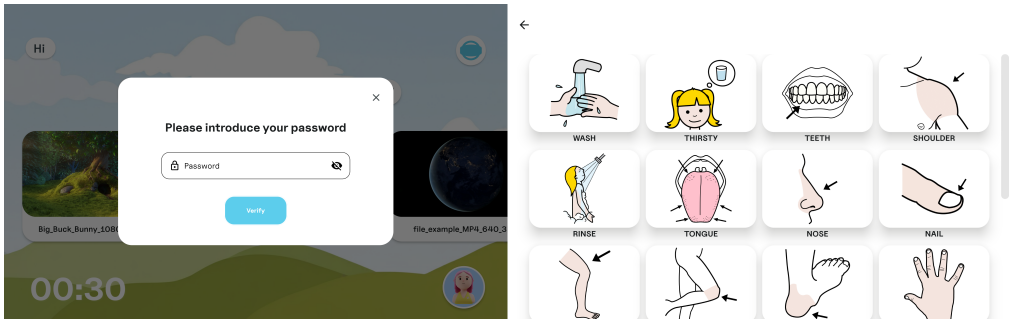


Figure 10.4: Password authentication & AAC communication panel [Own creation].

10.3 Settings Screen

10.3.1 Profile Settings Screen

Once we navigate to the settings screen, the first thing displayed is the user's information, including avatar, username, name, general statistics, and more, as shown in Figure 10.5.

- To change the current avatar, users can click the user avatar labeled as 1, where the app provides options to change the avatar from the gallery or the camera.
- To edit the username, users can click the edit icon labeled as 2. After making changes, a tick icon appears to confirm the update.
- To change the language, users can select the icon labeled as 3.
- The label 5 indicates the user's general statistics, while additional details can be accessed by clicking the expand icon labeled as 4.

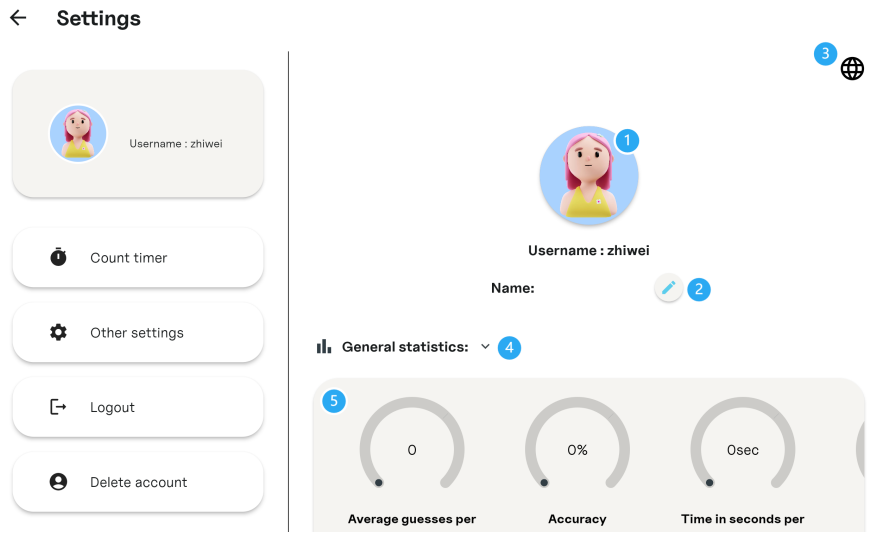


Figure 10.5: Profile settings screen [Own creation].

Figure 10.6 provides a clear representation of each action described above, with elements labeled accordingly for easy reference.

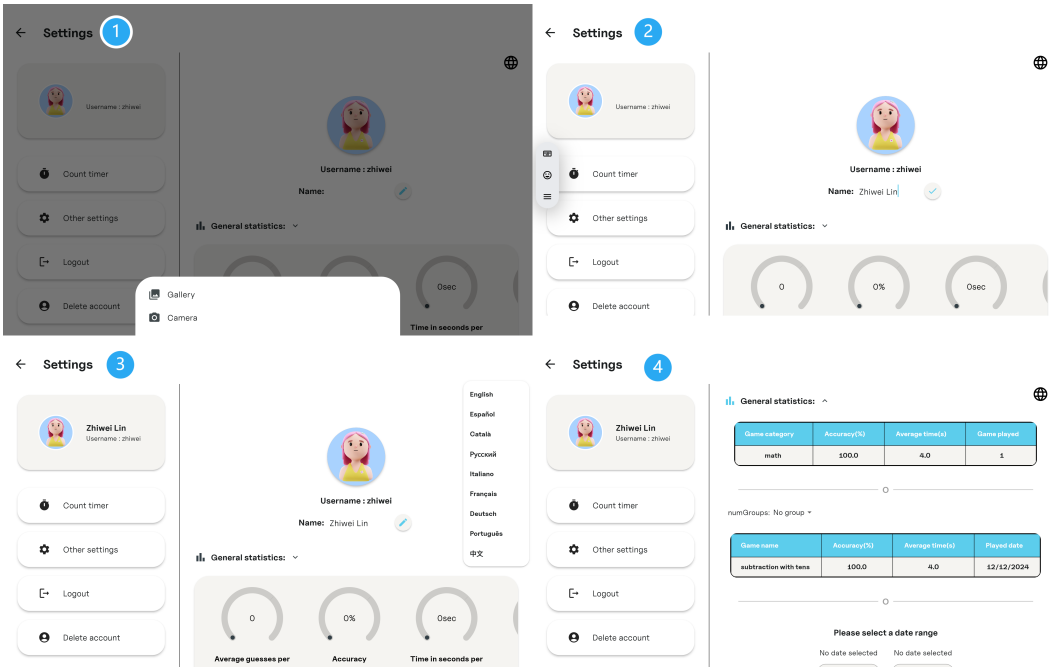


Figure 10.6: Overview of user actions on the profile screen [Own creation].

10.3.2 Timer Settings Screen

This screen is used to set the timer and determine the number of games the user needs to play once the timer ends. As shown in Figure 10.7:

- Label 1 represents the remaining time, which is displayed only when the timer is set.
- The minus and plus icons, labeled as 2 and 3, allow users to adjust the time by those icons or using the slider bar located between them. The selected time is displayed in label 4.
- To set the configured time, users need to click the button labeled as 6.
- If the users want to delete the timer, they just need to click the button labeled as 5.
- Finally, to set the number of the games the users need to play once the timer is finished refer to label 7, where yellow stars represent the number of games to be completed.

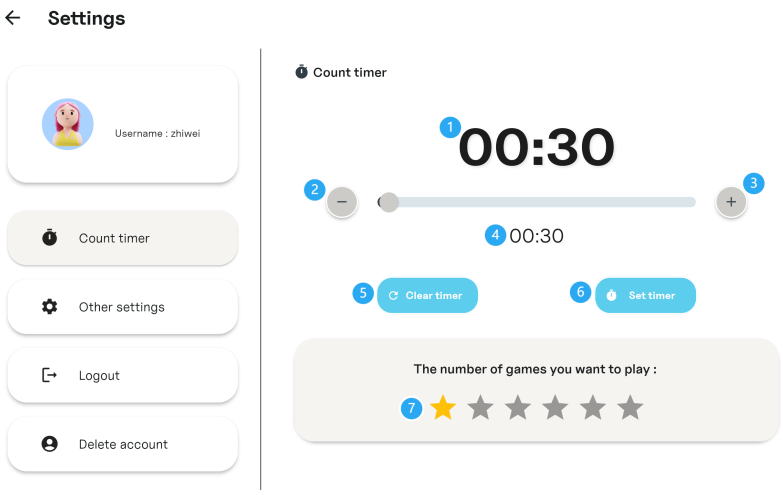


Figure 10.7: Timer settings screen [Own creation].

10.3.3 Other Settings Screen

The screen shown in Figure 10.8 contains all additional configurations, such as password requirement settings, game difficulty adjustment, blocking games, AAC panel cards, and more. The option labeled as number 2 allows users to toggle the visibility and accessibility of the communication panel option on the home screen (label 3 in Figure 10.3). The toggle option labeled as number 3 is used to enable or disable the password requirement for accessing the settings screen.

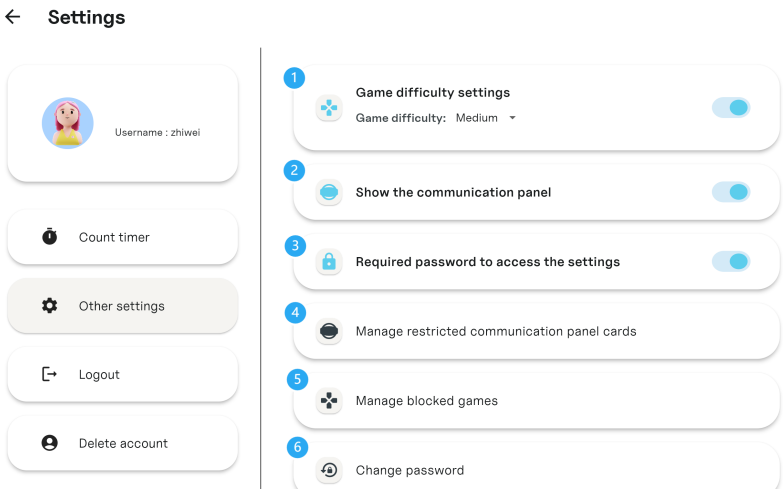


Figure 10.8: Other settings screen [Own creation].

Figure 10.9 shows the screens that appear when the options labeled as 1, 4, 5, and 6 in Figure 10.8 are selected.

- Label 1: This screen allows users to configure the game difficulty. There are 3 options: Easy, Medium, and Hard, the default option is Medium. Once the difficulty option is selected, only games of that level will be displayed. If the user disables the game difficulty settings, games will be displayed randomly.
- Label 4: This screen represents the management of AAC cards, where users can choose which panel cards will be displayed on the AAC communication screen. Users can block or unblock all panel cards by clicking the button in the bottom right corner.
- Label 5: This screen represents the management of games, where users can decide which games will be blocked and not displayed by clicking on a selected game.
- Label 6: This screen is for changing the password, where the user needs to enter a new password that is different compared to the original, and once entered, the user can decide to change it by clicking the change button.

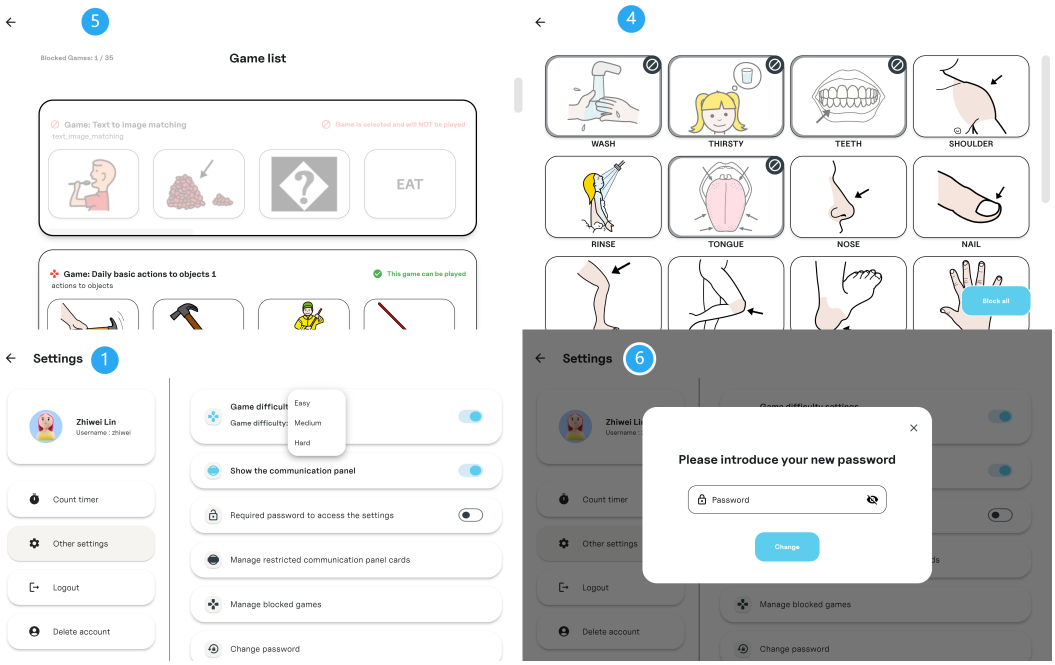


Figure 10.9: Detailed options for other settings [Own creation].

10.3.4 Logout and Delete Account Screens

Figure 10.10 represents the screens for the logout and delete account options. Both follow the same logic: to log out or delete the account, the user only needs to slide the slide bar, and both actions redirect to the login screen. However, there is an additional confirmation step to complete the account deletion, and once confirmed, it navigates to the login screen.

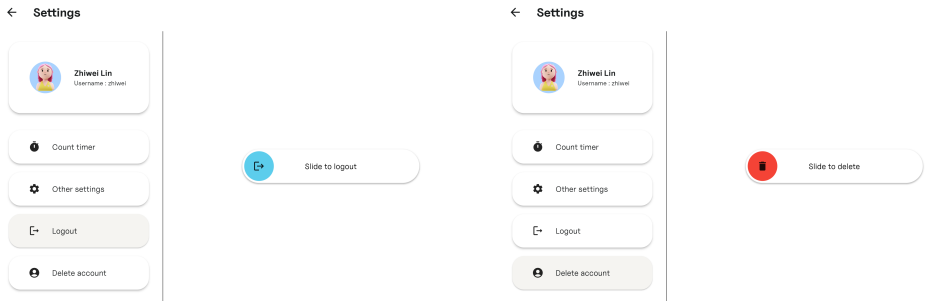


Figure 10.10: Logout and delete account [Own creation].

10.4 Admin Home Screen

This screen displays the admin home screen, as shown in Figure 10.11. In the red circle, there are two options: the admin option and the data export/import option, labeled as 1 and 2, respectively.

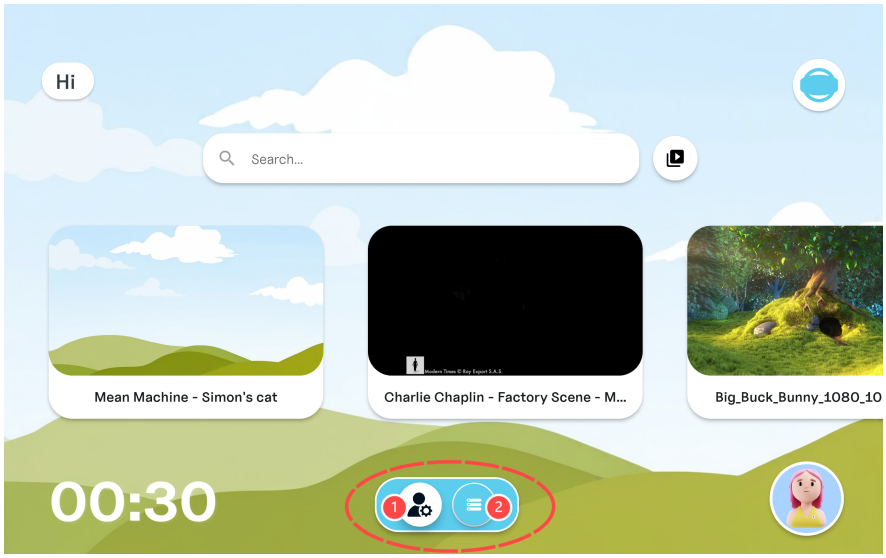


Figure 10.11: Admin home screen [Own creation].

- When clicking the admin option, the app navigates to the admin screen, which displays a list of games available and provides various operations related to games and the AAC panel (Figure 10.12).
- When selecting the data export/import option, the app displays various functionalities such as downloading a game file template, exporting games, importing games, and exporting the database (Figure 10.12).

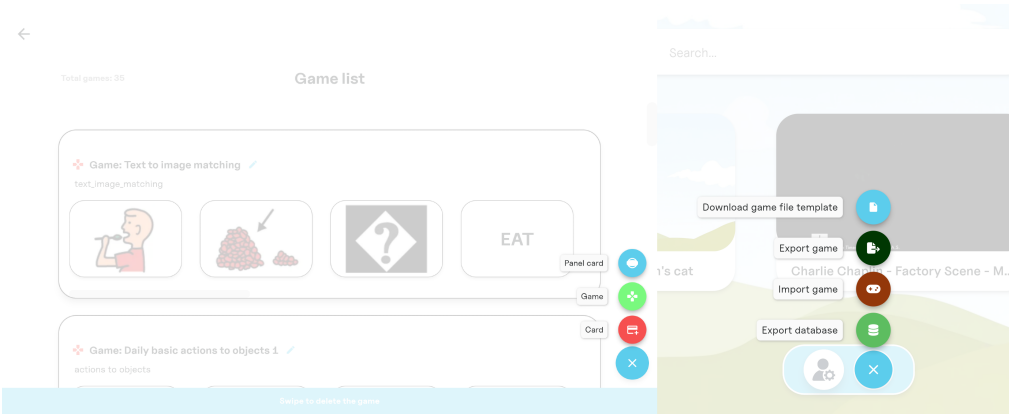


Figure 10.12: Admin option and data option [Own creation].

10.4.1 Data Export/Import Option

For the data export/import options:

- When the option to download a file template is selected, the app starts downloading the game template file named "game_data_template.txt" into the app, allowing the user to create a game in a more flexible way.
- When the import game option is selected, the app opens the file manager, where the user needs to select the game file to import.
- When the export database option is selected, the app starts to download the current database with the name "exported_database.zip" into the device.
- Finally, when the export game option is selected, the app displays a list of games on the screen (Figure 10.13), where the user needs to select the games to be exported. The selected games will be downloaded to the device as "exported_game_data.zip".

All exported data can be found in the **"Android/data/com.play_moment.app/files"** directory.

Export Game Screen

Figure 10.13 represents the export game screen, where users can select the games they want to export. Once the games are selected, users can click the "Export game" button to start downloading the selected games. In addition, users can click the "SELECT ALL" button to select all games.

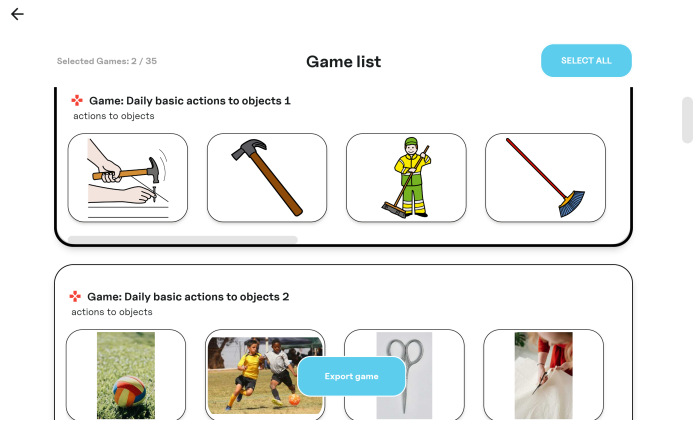


Figure 10.13: Export game screen [Own creation].

10.4.2 Admin Option

As mentioned previously, when the admin option is selected, the app displays a screen with a list of games (Admin screen). On this screen, users can view, edit, and delete games from the list. To delete a game, users only need to slide the selected game, which displays a confirmation dialog. Once confirmed, the selected game will be deleted from the database (Figure 10.14). Additionally, an option in the bottom right corner allows users to perform other operations, such as creating new games, as well as creating and deleting cards and AAC cards.

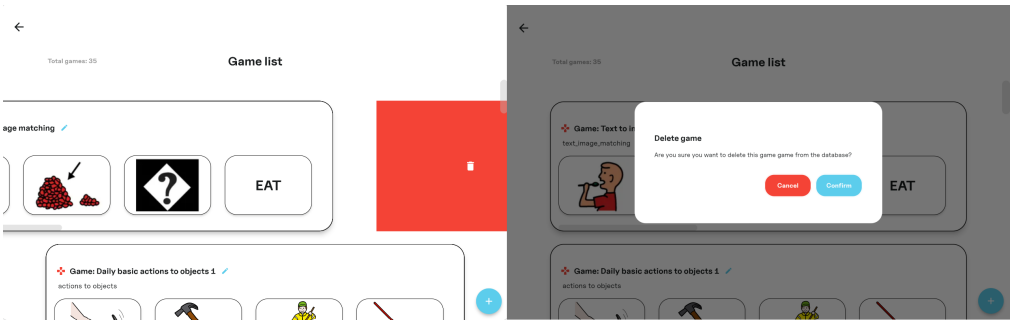


Figure 10.14: Delete game operation [Own creation].

Edit Game Screen

Figure 10.15 represents the edit game screen, which appears when the edit icon of Figure 10.12 is selected. Here, the admin user can update the game's information, such as its name, category, and the cards that are combined. To select a card, simply click on the desired card and the matched card will be selected automatically. Finally, once all changes are made, press the update button to update the game.

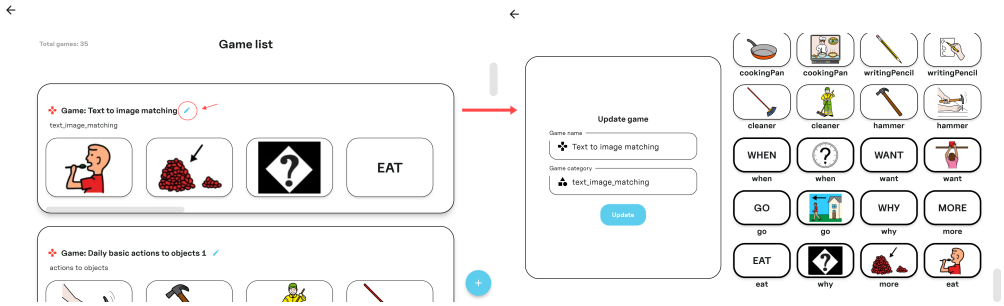


Figure 10.15: Edit game screen [Own creation].

Create Game Screen

To navigate to this screen (Figure 10.16), the Game option in the bottom right corner must be selected (Figure 10.12). To create a new game, users need to provide a unique game name and a category and choose a minimum of three pairs of cards. Once all the required information is set, users can press the create button to complete the game creation process.

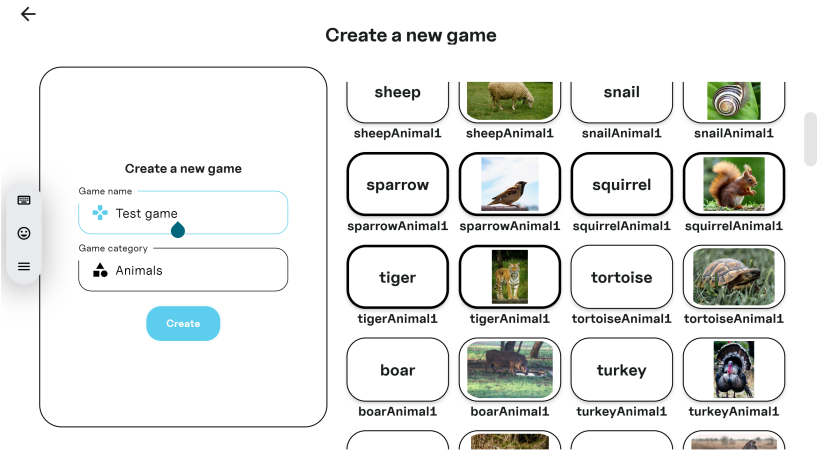


Figure 10.16: Create game screen [Own creation].

Create Card Screen

When the Card option in Figure 10.12 is selected, the Create Cards screen will be displayed (Figure 10.17). The text field labeled as 1 is for the matching card's name, which must be unique so that only two cards can share the same name. Once the card types are selected, they can be either Image or Text, as labeled in 2 and 3, and both can have different combinations.

- When the selected card type is **image**, an image must be provided in the image field (labeled as 4), along with an optional audio file (labeled as 5) that will be played on the play game screen when the card is dragged.
- When the selected card type is **text**, a text field (labeled as 6) is displayed, where the user needs to provide the text content that will appear on the card. Moreover, the text content will be read aloud as audio on the play game screen when the card is dragged.

Once all the information is filled, users need to press the "Create" button to create the matching card pair. The list of cards on the right will then refresh to include the newly created cards.

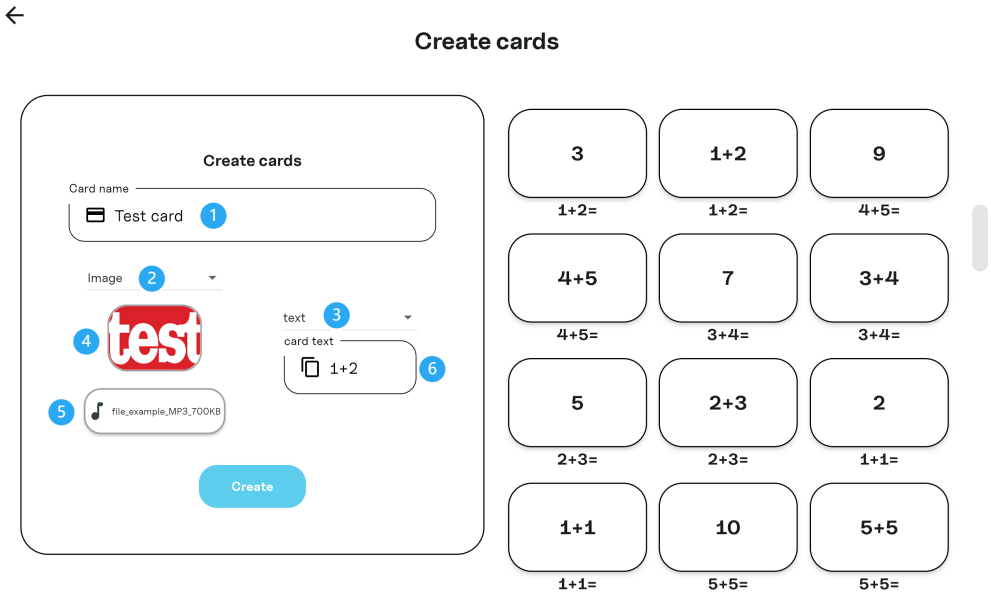


Figure 10.17: Create cards screen [Own creation].

Create AAC Card Screen

To navigate to this screen (Figure 10.18), users simply need to select the Panel card option in Figure 10.11. The text field labeled as 1 represents the name of the AAC card, while the voice input field specifies the text-to-speech (TTS) audio that will be produced when the AAC card is pressed. Finally, the field labeled as 3 is for the image, which will be presented on the card. All these fields are required, and once all the information is filled in by clicking the "Create" button, it will create the AAC card, and the AAC card list on the right will automatically refresh.

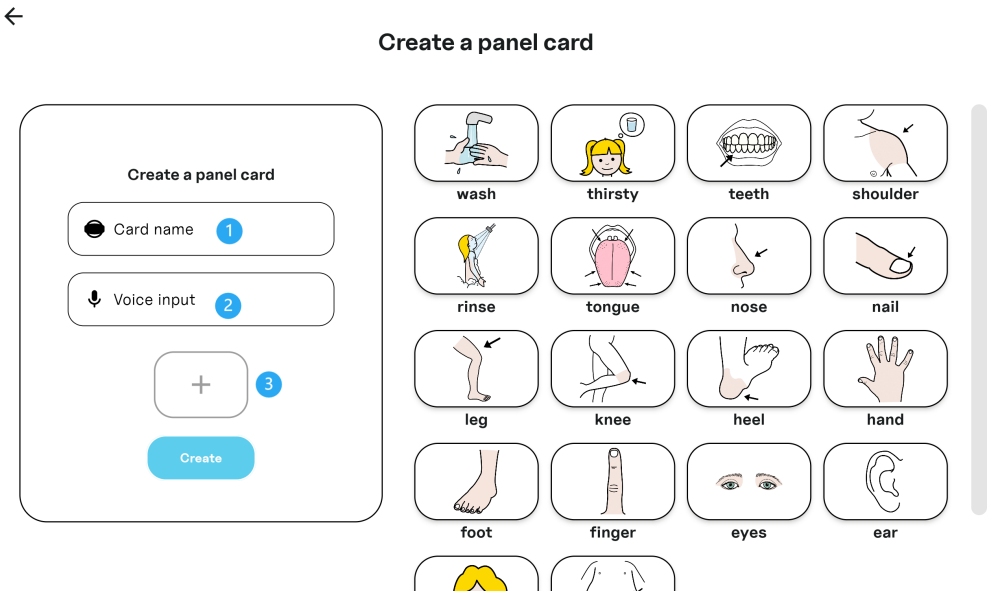


Figure 10.18: Create AAC card screen [Own creation].

Delete cards and AAC card

To delete cards or AAC cards, users need to long-press the selected card anywhere on the admin screen where the cards appear (Figure 10.19). This action will display a delete icon, allowing users to delete the selected card once confirmed.

If the deleted card is a non-AAC card (also known as a non-panel card), its matching card will also be deleted. Moreover, if the deleted card is part of any game, the corresponding game will be affected. If deleting the card results in the game having insufficient cards, the game will also be deleted. However, if the game is the last game in the app, the card deletion process will be canceled.

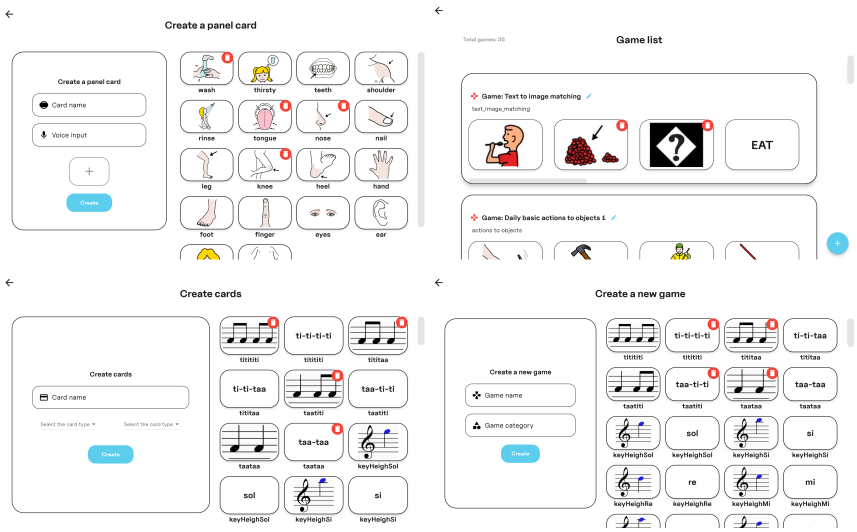


Figure 10.19: Delete cards and AAC card [Own creation].

10.5 Play Game Screen

This screen is displayed when the timer is finished. If the displayed game is being played for the first time by the user, the app provides an indication showing which card needs to be matched with another. The user can drag and drop cards to make a match, and if the cards match, the matched pair disappears (Figure 10.20).

Figure 10.20: Play game screen [Own creation].

Chapter 11

Legal Aspects

11.1 Laws and Regulations

11.1.1 Data Protection Law

Our application is fully offline, meaning that we do not have access to user data, since all information is stored locally on the user's device and can only be accessed by the user. However, it is essential to ensure the security and confidentiality of this data. In Spain, the Organic Law on Data Protection (**LOPD**) and the General Data Protection Regulation (**GDPR**) regulate the management of personal data.

To complete these regulations, the following principles need to be completed:

- **Purpose Limitation:** The data is stored only for application functionality and cannot be used for any other purpose [23].
- **Data accuracy:** User data must be accurate and updated, allowing users to modify it directly within the app [23].
- **User right:** Users have the right to delete, access, and modify their data. Since our application is offline, these actions need to be done by the user within the app or by uninstalling it [24].

11.1.2 Intellectual Property Rights

Since the project corresponds to Modality A and was developed under the supervision of a professor, the intellectual and industrial property rights are regulated by the regulations

approved by the Governing Council on 10/10/2008. According to these regulations, UPC holds ownership of the app while both the student and the professor are recognized as co-authors. If the UPC economically exploits the work, the authors are entitled to 50% of the net profits obtained [1].

11.2 Licenses

This project uses several Flutter libraries that are covered under open-source licenses, including **Apache 2.0**, **MIT**, and **BSD-3-Clause**. These licenses allow users to freely use, modify, and distribute the software. However, each one has specific requirements that must be included.

The images and audio files used in the app are licensed under various Creative Commons (CC) licenses, including Attribution 3.0, Attribution 4.0, and Creative Commons Zero (CC0). The default user avatar is provided by Freepik and is licensed under the Freepik license. The full details of the file licenses can be found in the "About" section of the app, on the profile settings screen.

11.2.1 BSD-3-Clause License

The BSD-3-clause License, also known as the "New BSD License" or "Modified BSD License," is applied to some of Flutter's native libraries and frameworks, such as flutter_localizations, cupertino_icons, shared_preferences, and others. To use these libraries, the following conditions must be met:

- The original copyright notice and the full text of the BSD-3-Clause License must be in the app.
- It is prohibited to use the name of the original copyright holder or contributor to promote the app or product without any permission.
- Users must be informed that the software is provided "as-is" with no warranty. This means that authors are not responsible for any damages or problems that arise.

11.2.2 MIT License

This license is applied to various Flutter libraries, such as audioPlayers, chewie, and more. It is a permissive open-source license, so users will have no problem using it freely. The requirements for this license are similar to those of the BSD-3-Clause license. We need to include the original copyright notice and the full license text in the app, along with a disclaimer that the software is provided "as-is" with no warranty.

11.2.3 Apache 2.0 License

This license is a bit more restricted compared to the BSD-3-Clause and MIT licenses. It requires including the copyright notice and the full license text. However, if any changes are made to the original code, we must clearly state what modifications we have made. Additionally, if the original author later files a patent lawsuit related to the software, the patent grant provided by the license will be revoked. Moreover, it is not allowed to use the name of the original author to promote the app, and it also includes the "as-is" with no warranty.

11.2.4 Creative Commons Zero License

This type of license is very flexible and permissive, allowing users to do anything with the work in any way, with no restrictions. Where the author has voluntarily waived all their rights and copyright. As a result, there is no need to mention the author, and the user is free to use it without any restriction.

11.2.5 Creative Commons Attribution 3.0 Unported License (CC BY 3.0)

This license allows users to share, modify, and distribute the work in any way. The only requirement is to give proper credit to the author of the work. This license works globally, but some terms may vary depending on the local law.

11.2.6 Creative Commons Attribution 4.0 International (CC BY 4.0)

Similar to CC BY 3.0, this license allows modifying and distributing the work as long as it gives credit to the original author. However, it has more clarity regarding moral rights and improved global coverage compared to CC BY 3.0.

11.2.7 Freepik License

The Freepik License is a permissive license that allows the use of free images and resources for both commercial and non-commercial use; the only requirement is to provide proper attribution in the credits section.

Chapter 12

Conclusions

12.1 Results and Discussion

The **PlayMoment** application now offers a remarkable learning experience for children with communication difficulties, helping them acquire knowledge, abilities, and various skills. It successfully combines learning with video entertainment, presenting different types of games that make the educational process engaging and enjoyable. This approach allows children to absorb information and knowledge comfortably and effectively.

This app is not only intended for personal use; it can also be utilized by different institutions and organizations. Using this application, they can save significant resources, both economically and environmentally, as it is reusable and eliminates the need for printed materials, making it an efficient solution that can benefit a large number of children.

In addition, the app is designed not just for the final thesis but also for long-term use. For this reason, it has been developed to function offline, significantly reducing maintenance costs. If the app were online, it would depend on a server and RESTful requests, requiring a continuously running server, which can be quite expensive, and once the thesis is completed, the server might no longer be maintained, making the app unusable.

By functioning offline, these issues are resolved, allowing the app to remain functional for a long period, until some point when devices are no longer compatible with the app. This makes the app highly usable and maintainable even in the distant future. Our goal is to support children with communication difficulties in their learning process, and the longer the app remains functional, the better it can fulfill its purpose and help us achieve our objectives.

Moreover, we also hope that in the future, as more users adopt this app, valuable feedback will be gathered to improve or expand its functionality, ensuring it can better address

users' needs. This will make the app more practical, functional, and useful. For this reason, the app's source code will also be shared for continued improvements.

12.2 Future Improvements

Although all of the initial objectives we set have been successfully achieved, there are still some areas for potential improvement. Below are some possible improvements:

- **Additional games:** Introduce more engaging and educational games to further improve the learning experience for children.
- **Search functionality for game creation and editing:** Add a search bar to the game creation and editing screens, making it easier to find and select the cards to include or remove from the games.
- **Video blocking:** Allow specific videos to be blocked from the video list for individual users, making them inaccessible, similar to the "block game" functionality.

12.3 Integration of Knowledge

During the project, a wide range of knowledge acquired from the Informatics Engineering degree was applied to develop the application. For instance:

- Regarding project management, including defining the scope, objectives, and analysis, the knowledge gained from **GPS** and **ER** played a significant role.
- The initial design of the mockups is based on the skills acquired in **GPS**, **ER**, and **PES**, where we became familiar with mockup design tools and understood the importance of having well-designed mockups.
- The initial design and specification of the UML diagrams and the database were influenced by concepts learned in **IES**, **AS**, **BD**, and **DBD**.
- The requirement analysis and use cases were based on methodologies introduced in **ER**, **GPS**, and **PES**, which helped us better organize ideas and define possible use scenarios.
- The programming languages used in the project were based on the knowledge gained from **PES**, where we had a deep understanding of that language. Moreover, the UI design principles were obtained from **IDI**.
- The descriptions and importance of testing, including unit testing, were derived from **PROP** and **PES**.

- The methodology used for the project development was acquired from **GPS**, where we explored different types of working methodologies for different scenes.
- Finally, the experience with version control using GIT was applied in **PES**, **ASW**, and **PROP**.

12.4 Achievement of Technical Competencies

- **CES1.1: To develop, maintain and evaluate complex and/or critical software systems and services. [In depth]**

  During this project, we developed an Android application using Flutter that meets all the objectives, as well as the functional and non-functional requirements of final users. The app is also scalable for future developers. In addition, we conducted thorough testing to ensure the app's reliability and robustness.

- **CES1.2: To solve integration problems in function of the strategies, standards and available technologies [In depth]**

  During the project, we resolved issues related to the integration between Flutter and the SQLite database. A JSON file was used to handle all the initial game data. In addition, we used the MVVM (Model-View-ViewModel) architectural pattern to ensure the communication between different components.

- **CES1.3: To identify, evaluate and manage potential risks related to software building which could arise. [In depth]**

  During the project, we identified potential risks that could arise, such as bugs, missed deadlines, and technological failures. We also developed solutions to address these problems.

- **CES1.4: To develop, maintain and evaluate distributed services and applications with network support. [A little bit]**

  For the application, we implemented a translation functionality that translates the content of text cards during their creation into all the languages supported by the app. This required integration with Google Translate.

- **CES1.5: To specify, design, implement and evaluate databases. [In depth]**

  For the app, we developed an SQLite database to manage all the information, including users, games, cards, and game played data. Moreover, we created a UML diagram to provide a clear representation of the database structure, including the corresponding primary and foreign keys.

- **CES1.6: To administrate databases (CIS4.3). [In depth]**

  The entire database is stored locally on the user's device, giving users complete control over their data. This ensures data security and privacy. Moreover, users can generate backups of their database if needed.

- **CES1.7: To control the quality and design tests in the software production. [Enough]**

  To ensure the app's functionality, proper operation, and UI component performance, we conducted various types of testing, such as unit testing, widget testing, and manual testing.

- **CES1.8: To develop, maintain and evaluate control and real-time systems. [A little bit]**

  For the app, we developed a timer system to control video pauses and manage the presentation of educational games.

- **CES1.9: To demonstrate the comprehension in management and government of software systems. [In depth]**

  During the development of the app, we established a clear plan that included all functionalities, objectives, tasks, and the working methodology, allowing us to track progress and manage the app efficiently.

- **CES2.1: To define and manage the requirements of a software system. [In depth]**

  Before starting the app development, we analyzed the needs of the end-users to define the requirements. During the app deployment we also collected user feedback to make improvements and adjustments.

- **CES2.2: To design adequate solutions in one or more application domains, using software engineering methods which integrate ethical, social, legal and economical aspects. [In depth]**

  All data is stored locally on the user's device, meaning no personal information is collected. In addition, the app reduces environmental impact by minimizing the need for printed materials, leading to significant cost savings. It also has no maintenance costs since it operates without the need for servers or online infrastructure. Additionally, it can benefit many children with communication difficulties by helping them integrate more easily into society.

- **CES3.1: To develop multimedia services and applications. [In depth]**

  The app allows users to play locally stored videos, which pause automatically when the timer ends to present educational games.

- **CES3.2: To design and manage a data warehouse. [In depth]**

  The app uses SQLite to store all the data, ensuring it is correctly stored and well-structured. Based on this data, the app can generate tables and charts representing user game performance and statistical information, including accuracy and progress development over time.

Appendix A

Initial Games of the App

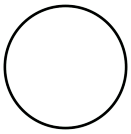
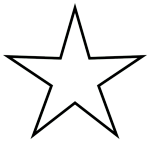
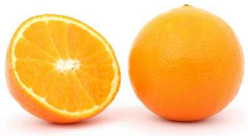
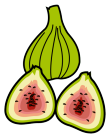
This appendix chapter describes all the initial game types prepared for the app. These games are specifically designed to improve various aspects of a child's abilities and skills.

The types of games are the following:

1. **Action-Object Matching**: Match an action with its associated objects (e.g., 'writing' with a pencil, 'cooking' with a pan).
2. **Actions Matching**: Match images of actions (e.g., running with another image of running). Use pictures of people performing different actions and match them with the same action image.
3. **PCS Shape Matching**: Match PCS symbols of geometric shapes (e.g., circle, square) with real-world images of objects in those shapes (e.g., clock, gift box).
4. **Number-Object Matching**: Match numbers with the correct quantity of objects (e.g., the number "3" with three apples).
5. **Color Matching**: Match objects with their corresponding colors (e.g., a red apple with a red square).
6. **PCS Object Matching**: Match PCS symbols with corresponding real-world objects (actions, food, animals).
7. **PCS Emotion Matching**: Match images of facial expressions showing different emotions (e.g., happy with happy, sad with sad) with the corresponding PCS symbol.
8. **Opposites Matching**: Match pairs of opposites (e.g., big with small, hot with cold).
9. **Calculation Matching**: Match math problems to their correct solutions (e.g., 40 + 40 with 80, 10 - 5 with 5).

10. **Musics symbols Matching**: Match musical rhythm symbols to their corresponding sounds (e.g., a 'sol' text to the 'sol' rhythm symbol).

In Table A.1 provides examples of all the game types mentioned previously.

Action-Object Matching			
			
Actions Matching			
			
PCS Shape Matching			
			
Number-Object Matching			
	1		3
Color Matching			
			
PCS Object Matching			
			

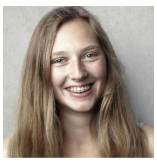
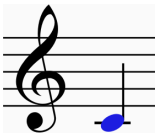
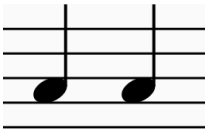
PCS Emotion Matching	
 	 
Opposites Matching	
sad happy	slow fast
Calculation Matching	
10+5 15	10-5 5
Musics symbols Matching	
 sol	 taa-taa

Table A.1: Game type examples [Own creation].

Appendix B

Use Case

Use Case	**Create account**
Primary Actor	Admin user
Precondition	The user is on the registration screen trying to create a new account.
Trigger	The user wants to create a new account within the app.
Main Scenario	1. The system requires the user to enter a username and password. 2. The user provides all required information. 3. The system validates the provided information (e.g., checks for duplicate usernames). 4. The system successfully creates the account, notifies the user, and redirects them to the login screen.
Extensions	2a. If any required field is left empty, the app displays an error message indicating the issue. 4a. Validation fails (e.g., the username already exists). 4a1. The app displays an error message indicating the issue. 4a2. The user corrects the information and resubmits the form.

Table B.1: Create account use case [Own creation]

Use Case	**Login**
Primary Actor	Admin user
Precondition	The user already has an account and is on the login screen.
Trigger	The user wants to access the app's functionalities.
Main Scenario	1. The app requests the user to enter their username and password. 2. The user enters the required credentials.

	3. The app validates the entered information (e.g., checks if the username exists and verifies the password). 4. If the credentials are correct, the app logs the user in and redirects them to the home screen.
Extensions	2a. If any required field is left empty, the app displays an error message indicating the missing information. 3a. If the credentials are incorrect, the app displays an error message. 3a1. The user re-enters their credentials and submits the form again.

Table B.2: Login use case [Own creation]

Use Case	**Login to admin account**
Primary Actor	Admin user
Precondition	There exists an admin account, and the user is on the login screen.
Trigger	The user attempts to access admin functionalities.
Main Scenario	1. The app requests the user to enter the admin username and password. 2. The user enters the required credentials. 3. The app validates the entered information (e.g., checks if the admin credentials are correct). 4. If the credentials are correct, the app logs the user in and redirects them to the admin home screen.
Extensions	2a. If any required field is left empty, the app displays an error message indicating the missing information. 3a. If the credentials are incorrect, the app displays an error message. 3a1. The user re-enters their credentials and submits the form again.

Table B.3: Admin account login use case [Own creation]

Use Case	**Change password**
Primary Actor	Admin user
Precondition	The user is logged in and is on the other settings option in the settings screen.
Trigger	The user wants to change the account password.
Main Scenario	1. The app displays a change password option on the settings screen. 2. The user selects the change password option in the settings screen. 3. The app prompts the user to enter the new password.

	4. The user enters the new password. 5. The app validates the new password, ensuring it is different from the old password. 6. The app updates the password and displays a message indicating that the password is changed successfully.
Extensions	5a. If the new password is the same as the old password, the app displays an error message. 5a1. The user needs to restart the process.

Table B.4: Change password use case [Own creation]

Use Case	**Logout**
Primary Actor	Admin user
Precondition	The user is logged in and is on the settings screen.
Trigger	The user wants to log out of the account.
Main Scenario	1. The app displays a logout option on the settings screen. 2. The user selects the logout option on the settings screen. 3. The app prompts the user to slide the logout bar from left to right to confirm logout. 4. The app closes the user session and redirects to the login screen.
Extensions	3a. If the user does not slide the bar completely from left to right, the logout action is canceled.

Table B.5: Logout use case [Own creation]

Use Case	**Delete regular account**
Primary Actor	Admin user
Precondition	The user is logged in and is on the settings screen.
Trigger	The user wants to delete the current account (not the admin account).
Main Scenario	1. The app displays a delete account option on the settings screen. 2. The user selects the delete account option on the settings screen. 3. The app prompts the user to slide the delete account bar from left to right to confirm account deletion. 4. The app requires the user to confirm the deletion. 5. The user confirms the deletion of the account. 6. The app deletes the user's account from the database and redirects to the login screen.
Extensions	3a. If the user does not completely slide the bar from left to right, the delete action is canceled.

	4a. If the user cancels the confirmation, the deletion will not be done.

Table B.6: Delete regular account use case [Own creation]

Use Case	**View profile**
Primary Actor	Admin user
Precondition	The user is logged in and is on the profile settings option in the settings screen.
Trigger	The admin user wants to view the logged-in user's performance and statistics information.
Main Scenario	1. The app displays the user's username, name, avatar, and general statistics. 2. By clicking the extend button next to the general statistics text, the app displays more detailed information about the user's statistics and performance.

Table B.7: View profile use case [Own creation]

Use Case	**Edit profile**
Primary Actor	Admin user
Precondition	The user is logged in and is on the profile settings option in the settings screen.
Trigger	The user wants to edit the profile information.
Main Scenario	1. The app displays the current user information. 2. The user edits the fields they want to change (e.g., name and avatar). 3. The app updates the user information.

Table B.8: Edit profile use case [Own creation]

Use Case	**Change language**
Primary Actor	Admin user
Precondition	The user is on the login screen or in the settings screen.
Trigger	The user wants to change the app language.
Main Scenario	1. The user selects the language option on the login screen or in the settings screen. 2. The user chooses the desired language from the list. 3. The app updates its content to the selected language.
Extensions	2a. If the user cancels the selection, the app retains the current language setting.

Table B.9: Change language use case [Own creation]

Use Case	Search video
Primary Actor	Child user
Precondition	The user is logged in, is on the home screen, has videos on their device, and has granted permission to access these videos.
Trigger	The user wants to search a video.
Main Scenario	1. The app displays a list of videos stored on the device and a video search bar on the top. 2. The user introduces the name of the video. 3. The app displays the videos that match the corresponding name.
Extensions	3a. If there is no video that matches the corresponding name, it displays the text "No video found".

Table B.10: Search video use case [Own creation]

Use Case	Play the video
Primary Actor	Child user
Precondition	The user is logged in, is on the home screen, has videos on their device, and has granted permission to access these videos.
Trigger	The user wants to watch a video.
Main Scenario	1. The app displays a list of videos stored on the device. 2. The user selects the video they want to watch. 3. The app plays the selected video and offers options to skip, pause, resume, and jump.
Extensions	2a. If the video format or codec is unsupported, the app displays an error message indicating this issue.

Table B.11: Play video use case [Own creation]

Use Case	Access to AAC functionality
Primary Actor	Child user
Precondition	The user is logged in and on the home screen.
Trigger	The user wants to use the AAC functionality.
Main Scenario	1. The app displays an AAC option button at the top right of the home screen. 2. The user selects the AAC functionality option. 3. The app displays a list of images with corresponding text labels. 4. The user clicks on the image that represents the idea they want to express. 5. The app produces the sound associated with the image's corresponding text.

Table B.12: Access to AAC functionality use case [Own creation]

Use Case	**Import video into the home screen**
Primary Actor	System User
Precondition	The user is logged in, is on the home screen, and the app has permission to access storage.
Trigger	Some videos on the device are not showing.
Main Scenario	1. The app displays an import video icon next to the video search bar. 2. The user clicks the import video button next to the search bar. 3. The user browses and selects the video files they want to import from the device storage. 4. The app displays a message indicating that the videos have been added to the home screen and asks if the user wants to play one of the imported videos. 5. The user confirms the video playback option. 6. The app plays the imported video.
Extensions	4a. If a video already exists on the home screen, the app displays a message about this problem and will not display the video playback option. 5a. If the user cancels the video playback option, the video will not be played.

Table B.13: Import video use case [Own creation]

Use Case	**Access to settings screen**
Primary Actor	Admin user
Precondition	The user is registered and is on the home screen.
Trigger	The user wants to access the configuration functionality.
Main Scenario	1. The app displays an option to access the settings screen. 2. The user clicks on the settings option. 3. The app displays a form and requests the user to enter the account password. 4. The app validates the entered password to check if it is correct. 5. If the password is correct, the app directs the user to the settings screen.
Extensions	4a. If the entered password is incorrect, an error message appears indicating this issue, and the user needs to restart the process.

Table B.14: Access settings screen use case [Own creation]

Use Case	**Set timer for video**
Primary Actor	Admin user

Precondition	The user is logged in and is on the timer settings option in the settings screen.
Trigger	The user wants to set a timer that will automatically display a game after a specific time of video playback.
Main Scenario	1. The app displays a configuration option for setting the timer and prompts the user to select the time interval using a slider or plus and minus icons. 2. The user sets the time interval for the timer and confirms it by clicking the "Set Timer" option. 3. The app sets the timer, displays a countdown on the screen, and shows a notifier. 4. When the user starts watching the video, the timer begins counting down. Once the timer reaches zero, the app displays the games.
Extensions	4a. If the timer is interrupted (e.g., if the app is closed) while the video is playing, the app saves its current state. When reopening, the app restores the timer state.

Table B.15: Set timer use case [Own creation]

Use Case	**Clear timer for video**
Primary Actor	Admin user
Precondition	The user is logged in and is on the timer settings option in the settings screen.
Trigger	The user wants to clear the timer.
Main Scenario	1. The app displays an option to clear the timer. 2. The user selects the reset option for the timer. 3. The app resets the timer and displays a message indicating that the timer has been canceled.
Extensions	2a. If the timer is not set, nothing will happen.

Table B.16: Clear timer use case [Own creation]

Use Case	**Set number of games to complete**
Primary Actor	Admin user
Precondition	The user is logged in and is on the timer settings option in the settings screen.
Trigger	The user wants to set a specific number of games that need to be completed when the timer ends.
Main Scenario	1. The app displays an option to configure the number of games to complete. 2. The user selects the number of games to be played. 3. The app saves the configuration and displays the number of games as stars.

Table B.17: Set number of games to complete use case [Own creation]

Use Case	**Block games**
Primary Actor	Admin user
Precondition	The user is logged in and clicked the managed blocked games option on the other settings option in the settings screen.
Trigger	The user wants to block certain games.
Main Scenario	1. The app displays a list of available games in the app. 2. The user selects the games that they want to block. 3. The app marks the blocked games in gray, indicating that these games cannot be played by the user.
Extensions	2a. If the game the user wants to block is the last unblocked game, the app displays an error message indicating this issue, and the game cannot be blocked.

Table B.18: Block games use case [Own creation]

Use Case	**Disable or enable AAC communication panel**
Primary Actor	Admin user
Precondition	The user is logged in and is on the other settings section in the settings screen.
Trigger	The user wants to enable or disable the AAC functionality within the app.
Main Scenario	1. The app presents an option to toggle the AAC functionality (enable/disable). 2. The user toggles the option to either enable or disable the AAC functionality. 3. The app updates the AAC functionality status based on the user's selection: • If enabled: The AAC option appears on the home screen. • If disabled: The AAC option disappears from the home screen.

Table B.19: Disable or enable the AAC communication panel use case [Own creation]

Use Case	**Block AAC communication cards**
Primary Actor	Admin user
Precondition	The admin user is logged in and is on the "Other Settings" section in the settings screen.
Trigger	The admin user wants to manage which communication panel cards will be displayed.
Main Scenario	1. The app presents an option to manage communication panel cards.

	2. The admin user selects this option. 3. The app displays all the available communication panel cards. 4. The admin user selects the card to be blocked. 5. The app displays the blocked card in grey color, with an icon indicating that it is blocked.

Table B.20: Block AAC communication cards use case [Own creation]

Use Case	**Disable or enable game difficulty settings**
Primary Actor	Admin user
Precondition	The admin user is logged in and is on the "Other Settings" section in the settings screen.
Trigger	The admin user wants to enable or disable the game difficulty setting within the app.
Main Scenario	1. The app presents an option to toggle the game difficulty setting (enable/disable). 2. The admin user toggles the option to either enable or disable the game difficulty settings. 3. The app updates the game difficulty settings status based on the user's selection: • If enabled: The user plays games based on the default difficulty and the user's progress. • If disabled: The user plays random games.

Table B.21: Disable or enable the game difficulty settings use case [Own creation]

Use Case	**Configure game difficulty**
Primary Actor	Admin user
Precondition	The admin user is logged in and in on the "Other Settings" section in the settings screen. The game difficulty setting is enabled.
Trigger	The admin user wants to configure the game difficulty to provide a better experience and learning process based on the child user's skills and progress.
Main Scenario	1. The app presents a selection bar to choose the game difficulty. 2. The admin user selects the desired game difficulty. 3. The app updates the game difficulty settings based on the user's selection.

Table B.22: Configure game difficulty use case [Own creation]

Use Case	**Enable or disable password requirement for settings screen**
Primary Actor	Admin user
Precondition	The user is logged in and is on the other settings section in the settings screen.

Trigger	The user wants to disable or enable the password requirement for accessing the settings screen.
Main Scenario	1. The app displays an option to toggle (enable/disable) the password requirement for the setting screen. 2. The user selects the option to either enable or disable the password requirement. 3. The app updates the security setting based on the user's choice: • If enabled: The app activates the password requirement to access the settings screen. • If disabled: The app deactivates the password requirement for the settings screen.

Table B.23: Enable/disable password requirement for settings use case [Own creation]

Use Case	**Import database**
Primary Actor	Admin user
Precondition	The user is not logged in and is on the login screen.
Trigger	The user wants to import a new database, replacing the current one.
Main Scenario	1. The app displays the option to import a database on the login screen. 2. The user selects the option to import the database. 3. If the app does not have permission to access device storage, it requests permission. 4. The app requires the user to confirm the import, warning that it will replace the current database. 5. The user confirms the import of the database. 6. The user selects the database to import. 7. The app verifies that the database is correct and imports the database. 8. If compatible, the database is imported, replacing the current one. 9. The app displays a message indicating the successful import of the database.
Extensions	3a. If the user grants the permission, it moves to step 4. 3b. If the user denies the permission, the database import is canceled. 4a. If the user cancels the confirmation, the import will not make any changes. 7a. If the selected database format is incompatible, the app displays an error message indicating the issue.

Table B.24: Import database use case [Own creation]

Use Case	Export database
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin home screen.
Trigger	The user wants to export the database for backup or other use.
Main Scenario	1. The app displays the option to export the database on the admin home screen. 2. The user selects the export database option. 3. If the app does not have permission to access device storage, it requests permission. 4. The app creates a copy of the current database in .db format and stores it on the device. 5. The app displays a message indicating that the database has been exported successfully.
Extensions	3a. If the user grants the permission, it moves to step 4. 3b. If the user denies the permission, the database exportation is canceled.

Table B.25: Export database use case [Own creation]

Use Case	Export game
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin home screen.
Trigger	The user wants to export games.
Main Scenario	1. The app displays the option to export games. 2. The user selects the export game option. 3. The app displays a list of games available for export and a button to export. 4. The user selects the games that they want to export and click the export button. 5. The app generates a zip file with all the selected games' info and downloads it to the device. 6. The app displays a message indicating the game has been successfully exported.
Extensions	4a. If the app does not have permission to access device storage, it requests permission. 4b. If the user grants permission, the process continues with step 5. 4c. If the user denies permission, the export is disabled. 5a. If no game is selected, the app displays an error message, so no zip file will be created.

Table B.26: Export game use case [Own creation]

Use Case	Import game

Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin home screen.
Trigger	The user wants to import a new game file into the app.
Main Scenario	1. The app displays the option to import games. 2. The user selects the import game option. 3. The app requests the user to select the game file to import. 4. The user selects the game file they want to import. 5. The app validates the imported game file. 6. If the file is valid, the app imports the game and adds it to the current game list. 7. The app displays a message indicating the game has been imported successfully.
Extensions	3a. If the app does not have permission to access device storage, it requests permission. 3b. If the user grants permission, the process continues with step 4. 3c. If the user denies permission, the import is disabled. 4a. If the user cancels the file selection, the process returns to the previous state. 6a. If there is an issue with the selected game file, the app displays an error message about this issue.

Table B.27: Import game use case [Own creation]

Use Case	**Download game file template**
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin home screen.
Trigger	The user wants to download a template game file to create new games in the correct format for the app.
Main Scenario	1. The app displays the option to download the game file template. 2. The user selects the download template option. 3. The app generates the template file in .txt format and saves it to the device. 4. The app notifies the user that the template has been successfully downloaded.
Extensions	2a. If the app does not have permission to access device storage, it requests permission. 2b. If the user grants permission, the process continues with step 3. 2c. If the user denies permission, the download is disabled.

Table B.28: Download game file template use case [Own creation]

Use Case	View the game list
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin home screen.
Trigger	The user wants to view all the games that the app has.
Main Scenario	1. The app displays the option to view the game list. 2. The user selects the admin screen option. 3. The app displays a list of all the games, showing their names, categories, and a list of the cards. 4. The user can scroll to view more games.

Table B.29: View game list use case [Own creation]

Use Case	Edit game
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin screen.
Trigger	The user wants to edit an existing game.
Main Scenario	1. The app displays a list of all the available games on the admin screen. 2. The user selects the game to edit from the list. 3. The app loads the game details, including the name, category, and the cards that are associated with the game. 4. The user modifies the necessary details (e.g., updates the name, category, or changes the cards). 5. The user updates the changes. 6. The app validates the updated game (ensuring a unique game name, a minimum of 6 cards, and that each card has its corresponding pair) and saves the changes. 7. The app notifies the user that the game has been successfully updated.
Extensions	4a. If the admin leaves required fields blank, the app displays an error message. 5a. If the admin cancels the editing process, no changes are made to the game. 6a. If there is an error saving the changes, the app displays an error message indicating the problem.

Table B.30: Edit game use case [Own creation]

Use Case	Delete game
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin screen.
Trigger	The user wants to delete an existing game.
Main Scenario	1. The app displays a list of all the available games on the admin screen.

	2. The user selects and slides the game to delete from the list. 3. The app prompts the user to confirm the deletion. 4. The user confirms the deletion. 5. The app deletes the selected game from the database and updates the game list. 6. The app notifies the user that the game has been successfully deleted.
Extensions	3a. If the user cancels the operation, the game will remain unchanged. 5a. If the selected game is the last game, an error message will appear, indicating that the game cannot be deleted.

Table B.31: Delete game use case [Own creation]

Use Case	**Create game**
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin screen.
Trigger	The user wants to create a new game.
Main Scenario	1. The app displays an option to create a new game on the admin screen. 2. The user selects the create game option. 3. The app presents a form for the user to enter the game name, category, and a list of cards that can be added. 4. The user fills in all the required data (e.g., game name, category, and cards). 5. The user saves the new game. 6. The app validates the game's data. If valid, it adds the game to the database. 7. The app notifies the user that the game has been created.
Extensions	4a. If the user leaves any required fields blank, the app displays an error message indicating the issue. 5a. If the user cancels the operation, no changes are made to the game. 6a. If there are any issues with the game data, the app will indicate the problem.

Table B.32: Create game use case [Own creation]

Use Case	**Create a new pair of cards**
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin screen.
Trigger	The user wants to create a new pair of card.
Main Scenario	1. The app displays an option to create a new pair of cards on the admin screen. 2. The user selects the create card option.

	3. The app presents a form for the user to enter card details such as name, card type, text content, voice input, or image. 4. The user fills in all the required data. 5. The user saves the new card. 6. The app validates the card's data. If valid, it adds the card to the database. 7. The app notifies the user that the card has been created.
Extensions	4a. If the user leaves any required fields blank, the app displays an error message indicating the issue. 5a. If the user cancels the operation, no changes are made to the card. 6a. If there is an issue with the card data, the app displays an error message indicating the problem.

Table B.33: Create a new pair of card use case [Own creation]

Use Case	**View cards**
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the card creation or management screen.
Trigger	The user wants to view all the existing cards.
Main Scenario	1. The app displays a list of existing cards on the card creation or management screen.

Table B.34: View cards use case [Own creation]

Use Case	**Delete card**
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on any screen displaying cards under the admin screen.
Trigger	The user wants to delete a pair of cards.
Main Scenario	1. The app displays a list of cards. 2. The user long presses the card they want to delete. 3. The app displays a delete icon in the top right corner of the card. 4. The user clicks the icon to delete. 5. The app shows a confirmation prompt to delete the card. 6. If confirmed, the app removes the selected card from the database, along with the matching pair card. 7. The app notifies the user that the card pair has been deleted.
Extensions	5a. If the user cancels the deletion, the cards will remain unchanged. 6a. If the selected card belongs to the last game, the app will notify the user of this issue.

Table B.35: Delete a pair of card use case [Own creation]

Use Case	Create a new AAC communication card
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the admin screen.
Trigger	The user wants to create a new AAC communication card.
Main Scenario	1. The app displays an option to create a new AAC communication card on the admin screen. 2. The user selects the create AAC panel card option. 3. The app presents a form for the user to enter AAC communication card details, such as name and voice input. 4. The user fills in all the required data. 5. The user saves the new AAC panel card. 6. The app validates the card's data. If valid, it adds the card to the database. 7. The app notifies the user that the AAC panel card has been created.
Extensions	4a. If the user leaves any required fields blank, the app displays an error message indicating the issue. 5a. If the user cancels the operation, no changes are made. 6a. If there are any issues with the data, the app will indicate the problem.

Table B.36: Create AAC communication card use case [Own creation]

Use Case	View AAC communication cards
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the create AAC communication card screen.
Trigger	The user wants to view all the existing AAC communication cards.
Main Scenario	1. The app displays a list of existing AAC communication cards on the card creation or management screen.

Table B.37: View AAC communication cards use case [Own creation]

Use Case	Delete an AAC communication card
Primary Actor	Admin user
Precondition	The user is logged in as admin and is on the create AAC communication card screen.
Trigger	The user wants to delete an AAC communication card.
Main Scenario	1. The app displays a list of AAC communication cards. 2. The user long presses the communication card they want to delete. 3. The app displays a delete icon in the top right corner of the card.

	4. The user clicks the icon to delete the card. 5. The app shows a confirmation prompt to delete the communication card. 6. If confirmed, the app removes the selected communication card from the database. 7. he app notifies the user that the selected communication card has been deleted.
Extensions	5a. If the user cancels the deletion, the cards will remain unchanged.

Table B.38: Delete an AAC communication card use case [Own creation]

Use Case	**Play the game**
Primary Actor	Child user
Precondition	The user is logged in and is on the game screen.
Trigger	The timer that was set has finished.
Main Scenario	1. The app displays a game formed by 3 pairs of cards, where the user has to match them. 2. The user drags the card they want to select and drops it onto another card. 3. If the match is correct, the matched cards will disappear.
Extensions	3a. If the match is incorrect, the cards will not change.

Table B.39: Play game use case [Own creation]

Appendix C

Test by Use Case

- **Create account**
  - **Description:** Create a new account with valid inputs.
    - **Objective:** Verify the user registration function to ensure that the user can create an account.
    - **Steps:**
      1. Navigate to the registration screen.
      2. Enter the corresponding username, password, and confirm password.
      3. Submit the form to create a new user.
      4. Navigate to the login screen.
      5. Enter the username and the password we just created.
      6. Submit the form.
    - **Expected results:**
      - The user is created successfully without any errors.
      - Any error message should be displayed.
      - After submitting the login form, the app navigates to the home screen.
    - **Actual results:**
      - The user is created successfully.
      - No error message is displayed.
      - After submitting the login form, the app navigates to the home screen.
  - **Description:** Create an account with invalid fields.
    - **Objective:** Verify that the app handles the invalid input.
    - **Steps:**
      1. Navigate to the registration screen.

2. Enter invalid inputs (e.g., a blank username, mismatched passwords, or a username that already exists).
3. Submit the form.

* **Expected results:**
  - "Username cannot be blank."
  - "Username already exists."
  - "Passwords do not match."
* **Actual results:**
  - All the expected error messages were displayed.
  - The form successfully submitted once the invalid input was fixed.

- **Log in**
  - **Description:** Test the login functionality with valid inputs.
    * **Objective:** Verify the login function.
    * **Steps:**
      1. Navigate to the login screen.
      2. Enter the username and the password.
      3. Submit the login form.
    * **Expected results:**
      - The user is logged in successfully without any errors.
      - Any error message should be displayed.
      - After submitting the login form, the app navigates to the home screen.
    * **Actual results:**
      - The user is logged in successfully.
      - No error message is displayed.
      - After submitting the login form, the app navigates to the home screen.
  - **Description:** Test the login functionality with invalid inputs.
    * **Objective:** Verify that the app handles the invalid inputs.
    * **Steps:**
      1. Navigate to the login screen.
      2. Enter invalid credentials.
      3. Submit the form.
    * **Expected results:**
      - The application should display the following error messages: "Invalid credentials."
      - The form should prevent submission until all errors are resolved.
    * **Actual results:**
      - All the expected error messages were displayed.
      - The form successfully submitted once the invalid input was fixed.

- **Log in to admin account**
  - **Description:** Test the login functionality of the admin account using valid credentials.
    - **Objective:** Verify that the admin can log in successfully.
    - **Steps:**
      1. Navigate to the login screen.
      2. Enter the admin username and the password (username: "admin", password: "admin").
      3. Submit the login form.
    - **Expected results:**
      - The admin is logged in successfully without any errors.
      - Any error message should be displayed.
      - After submitting the login form, the app navigates to the admin home screen.
    - **Actual results:**
      - The admin is logged in successfully.
      - No error message is displayed.
      - After submitting the login form, the app navigates to the admin home screen.
  - **Description:** Test the login functionality of the admin account using invalid credentials.
    - **Objective:** Verify that the app handles the invalid input.
    - **Steps:**
      1. Navigate to the login screen.
      2. Enter invalid credentials (e.g., username: "admin", password: "administration").
      3. Submit the form.
    - **Expected results:**
      - The application should display the following error messages: "Invalid credentials."
      - The form should prevent submission until all errors are resolved.
    - **Actual results:**
      - All the expected error messages were displayed.
      - The form successfully submitted once the invalid input was fixed.
- **Change password**
  - **Description:** Test the feature of changing the password.
    - **Objective:** Verify that the password can be changed successfully.
    - **Steps:**

1. Log in to the user account.
2. Navigate to the "Change password" section.
3. Enter a valid new password.
4. Confirm the password.
5. Log out and then log in using the new password.

* **Expected results:**
  - The password is changed correctly.
  - The user can log in with the new password, and the old password is no longer valid.
* **Actual results:**
  - The password is changed, and we can only log in with the new password.

– **Description:** Test the error handling when the new password is the same as the original.
* **Objective:** Verify that the app displays an error message if the new password matches the current one.
* **Steps:**
  1. Log in to the user account.
  2. Navigate to the "Change password" section.
  3. Enter an invalid password.
  4. Confirm the password.
* **Expected results:**
  - The app displays an error message indicating the same password error.
* **Actual results:**
  - The app displays the same password error.

- **Log out**
  – **Description:** Test the logout feature.
  * **Objective:** Verify that the user can log out and redirect to the login screen.
  * **Steps:**
    1. Log in to the user account.
    2. Navigate to the "Log out" section.
    3. Confirm the logout.
  * **Expected results:**
    - The user is logged out and redirected to the login screen.
  * **Actual results:**
    - The user is logged out and redirected to the login screen.

- **Delete regular account**
  – **Description:** Test the delete account functionality.

    - **Objective:** Verify that the user can be deleted from the database.
    - **Steps:**
      1. Log in to the user account.
      2. Navigate to the "Delete account" section.
      3. Confirm the deletion.
    - **Expected results:**
      - The user is deleted from the database and redirects to the login screen.
    - **Actual results:**
      - The user is deleted from the database and cannot be recovered.
      - Redirect to the login screen.

- **View profile**
  - **Description:** To test that the app displays the user's information correctly.
    - **Objective:** Verify that the user's information can be displayed.
    - **Steps:**
      1. Log in to the user account.
      2. Navigate to the "User profile" section.
    - **Expected results:**
      - The user's profile information is displayed correctly (e.g., username, name, avatar).
      - The user's statistics information is also shown successfully (e.g., the user's general statistics, such as accuracy, number of games, historical record, performance according to time).
    - **Actual results:**
      - All the user's information is shown correctly.

- **Edit profile**
  - **Description:** Test the ability of editing the user's profile information.
    - **Objective:** Verify that the user's information can be changed.
    - **Steps:**
      1. Log in to the user account.
      2. Navigate to the "User profile" section.
      3. Edit the editable field (e.g., name and avatar).
      4. Save the changes.
    - **Expected results:**
      - The profile information is updated.
    - **Actual results:**
      - The corresponding user's profile information is changed correctly.

- **Change language**

  - **Description:** Test the feature of changing the language of the application on the settings screen.
    * **Objective:** Verify that the app's language can be changed.
    * **Steps:**
      1. Log in to the user account.
      2. Go to the settings screen.
      3. Choose the new language.
    * **Expected results:**
      · The selected language is applied.
      · All the elements reflect the changes made.
    * **Actual results:**
      · All the app content reflects the selected language.

  - **Description:** Test the feature of changing the language of the application on the login screen.
    * **Objective:** Verify that the app's language can be changed.
    * **Steps:**
      1. Navigate to the login screen.
      2. Choose the language.
    * **Expected results:**
      · The selected language is applied.
      · All the elements reflect the changes made.
    * **Actual results:**
      · All the app content reflects the selected language.

- **Search video**
  - **Description:** Test the search function.
    * **Objective:** Verify that the video searching function is working correctly, helping the user to find the desired video.
    * **Steps:**
      1. Log in to the user account.
      2. Introduce the name of the video in the search bar.
      3. Verify that the displayed videos match the search query.
    * **Expected results:**
      · The videos shown match the entered search name.
    * **Actual results:**
      · The videos correspond to the search query.

- **Play video**
  - **Description:** Test the video playback feature.

    * **Objective:** Verify that the video playback functionality is working correctly.
    * **Steps:**
        1. Log in to the user account.
        2. Select the video we want to play.
        3. Verify the video starts to play.
        4. Check the video controls (e.g., pause, volume, skip).
        5. Wait for the timer to end.
        6. Verify that the video pauses automatically when the timer ends and displays educational games.
        7. Once the game finishes, verify that the video starts to play automatically.
    * **Expected results:**
        · The video controls work correctly.
        · The video plays without any problem.
        · The video resumes automatically once the game is finished.
        · The video pauses automatically once the timer is ends.
    * **Actual results:**
        · The corresponding functionality works correctly without any problem.

- **Access to AAC functionality**
  - **Description:** Test the AAC panel functionality.
    * **Objective:** Verify that the AAC communication panel functions correctly and produces the corresponding audio based on the app's language.
    * **Steps:**
        1. Log in to the user account.
        2. Select the AAC communication panel option.
        3. Click on the card we want to play.
        4. Verify that the card produces the corresponding audio.
        5. Go to the language settings and change the app's language.
        6. Repeat steps 2–4.
    * **Expected results:**
        · The card produces the corresponding audio even though we changed the app's language.
    * **Actual results:**
        · The selected card produces the corresponding audio under different app language settings.

- **Import video into the home screen**
  - **Description:** Import a video that is not on the video list.
    * **Objective:** Verify that the video can be imported successfully.

    * **Steps:**
        1. Log in to the user account.
        2. Click on the import video option.
        3. Select a video file to upload.
        4. Wait for the video to be imported.
        5. Verify that the video is imported.
    * **Expected results:**
        · The app displays a message indicating that the video is imported.
        · The video is imported.
        · The app displays a message indicating if the user wants to play the imported video.
    * **Actual results:**
        · The video is uploaded successfully and appears on the video list.
        · The imported message is displayed.
        · The dialog asking to play the imported video appeared.

  – **Description:** Import a video that is on the video list.
    * **Objective:** Verify that the video cannot be imported.
    * **Steps:**
        1. Log in to the user account.
        2. Click on the import video option.
        3. Select a video file to upload.
        4. Wait for the video to be imported.
        5. Verify that the video is not imported.
    * **Expected results:**
        · The app displays an error message indicating that the video is already imported.
        · The app displays a message indicating if the user wants to play the imported video.
    * **Actual results:**
        · The error message is displayed.
        · The dialog asking to play the imported video appeared.

- **Access to settings screen**
  – **Description:** Verify access to the settings screen with a password requirement.
    * **Objective:** Ensure that access to the settings screen requires a valid password for authentication.
    * **Steps:**
        1. Log in to the user account.
        2. Select the settings screen option.

3. When prompted, enter the password associated with the account.
4. Verify that the correct password redirects the user to the settings screen.
5. Verify that incorrect password triggers an error message and prevents access.

* **Expected results:**
  - A password is required to enter the settings screen.
  - Upon entering the correct password, the settings screen is displayed.
  - If an incorrect password is entered, the app displays an error message and does not allow access to the settings screen.
* **Actual results:**
  - The password requirement is needed.
  - Entering an incorrect password prevents access, and the error message appears correctly.
  - When the password is correct, the settings screen is displayed correctly.

- **Set timer for video**
  - **Description:** Verify that the timer can be set and modified correctly for the video playback.
    * **Objective:** Ensure that the timer works as expected and responds appropriately during video playback.
    * **Steps:**
      1. Log in to the user account.
      2. Navigate to the "Count timer" section.
      3. Configure the desired time.
      4. Navigate to the home screen.
      5. Select a video to play.
      6. wait for few seconds to observe the timer.
      7. Return to the home screen.
      8. Verify that the timer counts down correctly.
    * **Expected results:**
      - The timer can be configured and set correctly.
      - A message is shown once the timer is set.
      - The timer starts to count down when the video playback begins.
      - The timer stops once the user returns to the home screen from video playback.
    * **Actual results:**
      - The confirmation message is displayed after the timer is set.
      - The timer is displayed and functions correctly.
      - The timer begins counting down only when the video starts playing.
      - The timer pauses when returning to the home screen, as expected.

- **Description:** Verify that the timer triggers the educational games.
  - **Objective:** Ensure that the educational games are triggered automatically when the timer finishes and that the timer resets as expected.
  - **Steps:**
    1. Log in to the user account.
    2. Navigate to the "Count timer" section.
    3. Configure the desired time.
    4. Navigate to the home screen.
    5. Select a video to play.
    6. Wait until the timer finishes counting down.
    7. Verify that the game is displayed immediately after the timer is finished.
    8. Complete the educational game and return to the home screen.
    9. Repeat steps 5-7 to verify that the timer is working correctly.
  - **Expected results:**
    - The educational game is launched automatically when the timer is finished.
    - After the educational game is completed, the timer resets with the previously set time.
    - The timer starts to count down only when the video is being played.
  - **Actual results:**
    - The timer works correctly and starts to count down only during video playback.
    - The educational game is present when the timer is finished.
    - The timer resets automatically with the previously defined time.

- **Description:** Verify that the timer can be set without needing to first clear it.
  - **Objective:** Ensure that the timer works as expected.
  - **Steps:**
    1. Log in to the user account.
    2. Navigate to the "Count timer" section.
    3. Configure the desired time.
    4. Navigate to the home screen.
    5. Select a video to play.
    6. wait for few seconds.
    7. Return to the home screen.
    8. Verify that the timer counts down correctly.
    9. Repeat steps 2-8 to verify that the timer can be set again without clearing it first.
  - **Expected results:**
    - The timer can be configured and set correctly.

        - A message is shown once the timer is set.
        - The timer starts to count down when the video playback begins.
        - The timer stops once the user returns to the home screen from video playback.
      - **Actual results:**
        - The confirmation message is displayed after the timer is set.
        - The timer is displayed and functions correctly.
        - The timer begins counting down only when the video starts playing.
        - The timer pauses when returning to the home screen, as expected.

- **Clear timer for video**
  - **Description:** Verify that the timer can be deleted.
    - **Objective:** Ensure that the timer can be deleted successfully and does not trigger any games afterward.
    - **Steps:**
      1. Log in to the user account.
      2. Navigate to the "Count timer" section.
      3. Delete the timer.
      4. Navigate to the home screen.
      5. Select a video to play.
      6. Wait for a few seconds.
      7. Return to the home screen.
      8. Verify that the timer is not active.
    - **Expected results:**
      - The timer is deleted successfully.
      - A message is displayed once the timer is deleted.
      - No timer-related educational games are triggered.
    - **Actual results:**
      - The message confirming the deletion of the timer is shown.
      - The timer is deleted correctly and no games are triggered.

- **Set number of games to complete**
  - **Description:** Verify that the user can set the number of games to be completed.
    - **Objective:** Ensure that the number of games that need to be completed once the timer is finished is set correctly.
    - **Steps:**
      1. Log in to the user account.
      2. Navigate to the "Count timer" section (Timer already set).
      3. Select the number of games to be completed.
      4. Navigate to the home screen.

5. Select a video to play.
6. Wait until the timer is finished.
7. Verify that the number of games to be played corresponds to the number we have set.

* **Expected results:**
  · The number of games is set correctly.
* **Actual results:**
  · The number of games is set correctly, no matter how many times the timer finishes.

- **Block games**
  - **Description:** Block some games.
    * **Objective:** Ensure that only the games that are not blocked can be played once the timer is finished.
    * **Steps:**
      1. Log in to the user account.
      2. Navigate to the "Manage blocked games" section.
      3. Select the games to be blocked.
      4. Return to the home screen.
      5. Select a video to play.
      6. Wait until the timer is finished.
      7. Verify that only the unblocked games are displayed.
    * **Expected results:**
      · The blocked games are not displayed.
    * **Actual results:**
      · Only the unblocked games are shown.
  - **Description:** Try to block all educational games.
    * **Objective:** Ensure that it is not possible to block all the games and that at least one game remains unblocked.
      1. Log in to the user account.
      2. Navigate to the "Manage blocked games" section.
      3. Try to block all available games.
    * **Expected results:**
      · The app displays an error message indicating that blocking all games is not allowed.
      · Users are only permitted to block up to n-1 games, where n is the total number of available games.
    * **Actual results:**
      · An error message is displayed when trying to block all games.

        - The user is only able to block up to n-1 games, ensuring that at least one game is unblocked.

- **Disable or enable AAC communication panel**
  - **Description:** Make the AAC communication panel accessible or not.
    - **Objective:** Verify that the AAC communication panel can be enabled or disabled based on the user's decision.
    - **Steps:**
      1. Log in to the user account.
      2. Navigate to the "Other settings" section where AAC communication panel toggle is located.
      3. Enable the AAC communication panel.
      4. Navigate to the home screen.
      5. Verify that the AAC communication panel is accessible and functional.
      6. Repeat step 2.
      7. Disable the AAC communication panel.
      8. Navigate to the home screen.
      9. Verify that the AAC communication panel is no longer accessible and visible.
    - **Expected results:**
      - When the user enables the AAC panel, it becomes accessible and visible.
      - When the user disables the AAC panel, it becomes inaccessible an invisible.
    - **Actual results:**
      - The AAC accessibility is working correctly based on the user's decision.

- **Block AAC communication cards**
  - **Description:** To decide which communication cards are visible in the AAC communication panel.
    - **Objective:** Ensure that the blocked AAC communication cards are not visible.
    - **Steps:**
      1. Log in to the user account.
      2. Navigate to the "Manage restricted communication panel cards" option.
      3. Select the cards to be blocked.
      4. Verify that the blocked cards are not visible in the communication panel.
      5. Verify that all non-blocked cards are visible and functioning correctly.
    - **Expected results:**
      - The selected card is not visible.

        - All the non-blocked cards are working correctly.
    - **Actual results:**
        - The blocked panel cards are not displaying, and others are showing and working correctly.

- **Disable or enable game difficulty settings**
    - **Description:** Verify that the user can enable or disable game difficulty settings.
        - **Objective:** Ensure that games displayed are accurate based on the user's decision.
        - **Steps:**
            1. Log in to the user account.
            2. Navigate to the "Other settings" section where the game difficulty settings toggle is located.
            3. Enable the game difficulty settings.
            4. Verify that the game displayed is based on the default game difficulty (medium).
            5. Disable the game difficulty settings.
            6. Verify that games are displayed randomly without considering difficulty.
        - **Expected results:**
            - When the user enables the game difficulty setting, the difficulty option is visible, and the default game difficulty is set to "Medium."
            - When the user disables the game difficulty settings, the difficulty option is no longer visible.
            - Games are displayed according to the user's selection when the setting is enabled and randomly when disabled.
        - **Actual results:**
            - The actual results are the same as the expected results.

- **Configure game difficulty**
    - **Description:** Configure the difficulty of the game.
        - **Objective:** Ensure that the user can select a game difficulty, and the games displayed align with the chosen difficulty.
        - **Steps:**
            1. Log in to the user account.
            2. Navigate to the "Other settings" section where the game difficulty settings toggle is located.
            3. Enable the game difficulty settings.
            4. Select the difficulty (e.g., Easy, Medium, Hard).
            5. Verify that the games displayed correspond to the selected difficulty.
        - **Expected results:**

        - The user can select the desired game difficulty.
        - The games displayed match the selected difficulty level.
      - **Actual results:**
        - The game difficulty can be selected, and the displayed games correspond to the chosen difficulty level.

- **Enable or disable password requirement for settings screen**
  - **Description:** Set up the password requirement for accessing the settings screen.
    - **Objective:** Ensure that the password requirement can be enabled or disabled.
    - **Steps:**
      1. Log in to the user account.
      2. Navigate to the "Password requirement for settings screen" section.
      3. Disable the password requirement.
      4. Return to the home screen and then navigate to the settings screen.
      5. Verify that the settings screen can be accessed without a password.
      6. Repeat step 2.
      7. Enable the password requirement.
      8. Repeat step 4.
      9. Verify that a password is required to access the settings screen.
    - **Expected results:**
      - When the user enables the settings requirement, accessing the settings screen requires a password.
      - When the user disables the settings requirement, accessing the settings screen does not require a password.
    - **Actual results:**
      - The enable and disable functions of the password requirement for accessing the settings screen work correctly.

- **Import database**
  - **Description:** Import the database into the device.
    - **Objective:** Ensure that the database can be imported successfully to the device.
    - **Steps:**
      1. Navigate to the login screen.
      2. Select the import database option.
      3. Confirm the database import.
      4. Select the appropriate database zip file.
      5. Start the import process.
      6. Log in with the imported database username and password.

7. Verify that all data has been successfully imported.

* **Expected results:**
  - The app should display a message confirming the database is imported.
  - The database should be imported without any issues.
* **Actual results:**
  - The confirmation message appears correctly.
  - The database is imported successfully with the correct data.

– **Description:** Try to import an incorrect database file.

* **Objective:** Ensure that the app handles different import scenarios.
* **Steps:**
  1. Navigate to the login screen.
  2. Select the "Import database" option.
  3. Confirm the database import.
  4. Select an incorrect database zip file.
  5. Start the import process.
  6. Verify that an error message appears indicating the problem with the file.
* **Expected results:**
  - The app should display a clear error message when the selected file is invalid.
* **Actual results:**
  - The error message is shown correctly.

- **Export database**

  – **Description:** Export the database to the device.

  * **Objective:** Ensure the database is successfully downloaded to the device without any issues.
  * **Steps:**
    1. Log in to the admin account.
    2. Navigate to the "Export database" section.
    3. Wait for the export process to complete.
    4. Access the Android device's file system.
    5. Navigate to the Android directory.
    6. Open the app's file folder to find a zip file that contains the database and related files.
  * **Expected results:**
    - The app displays a message indicating that the database is exported.
    - The exported database is located in the correct folder.
    - The zip file contains the database information and necessary files.

    - **Actual results:**
        - The message is showing correctly.
        - The database zip file is downloaded successfully to the appropriate folder.
        - The downloaded database files are accurate.

- **Export game**
    - **Description:** Export selected games to the device.
        - **Objective:** Ensure that the selected games are successfully downloaded to the device.
        - **Steps:**
            1. Log in to the admin account.
            2. Navigate to the "Export games" section.
            3. Select the games to be exported.
            4. Download the selected games.
            5. Wait for the export process to complete.
            6. Access the Android device's file system.
            7. Navigate to the Android directory.
            8. Open the app's file folder to find a zip file that contains the selected games' data.
        - **Expected results:**
            - The app displays a message indicating that the games are exported.
            - The games can be found in the corresponding folder.
            - The zip file contains all the game's data.
        - **Actual results:**
            - The message is showing correctly.
            - The game zip file is downloaded correctly and is in the corresponding folder.
            - The downloaded game files are correct.
    - **Description:** Export all the games.
        - **Objective:** Ensure that the selected games are downloaded to the device.
        - **Steps:**
            1. Log in to the admin account.
            2. Navigate to the "Export games" section.
            3. Select all games by clicking the select all option.
            4. Download the selected games.
            5. Wait for the export process to complete.
            6. Access the Android device's file system.
            7. Navigate to the Android directory.

8. Open the app's file folder, where a zip file that contains the game data is located.

* **Expected results:**
  - The app displays a message indicating that the games are exported.
  - The games can be found in the corresponding folder.
  - The zip file contains all the games' data.
* **Actual results:**
  - The message is showing correctly.
  - The game zip file is downloaded correctly and is in the corresponding folder.
  - The downloaded game files are correct.

- **Import game**
  - **Description:** Import various types of games.
    * **Objective:** Ensure that the app can handle different game import situations.
    * **Steps:**
      1. Log in to the admin account.
      2. Navigate to the "Import games" section.
      3. Select the file to be imported, which contains different game scenarios (e.g., games with the same name but different ID, games with the same ID but different name, games with identical name and ID, games with different names and IDs, and games containing cards with similar combinations of names and IDs).
      4. Wait for the import process to complete.
      5. Verify that the games and associated cards are imported correctly.
    * **Expected results:**
      - The app displays a message indicating that the games are imported.
      - The game is imported correctly to the database.
      - When a game with the same name but different ID is imported, the existing game is renamed using the format "originalName_number".
      - When a game with the same ID but different name is imported, the imported game is assigned a new available ID.
      - When a game with the same ID and same name is imported, only the new cards that are not part of the current game are added.
      - When a game with a different name and ID is imported, it is imported successfully without issues.
      - When cards in the imported game have the same ID and name, the card will not be added again.
      - When cards in the imported game have the same name but different IDs, the original cards are renamed as "originalName_number."

        - When the cards in the imported game are new, they are imported successfully.
        - When the cards in the imported game have the same ID but different names, the new cards are assigned different IDs.
      - **Actual results:**
        - The success message appears correctly.
        - All types of games are imported successfully.

  - **Description:** Try to import an incorrect game file.
    - **Objective:** Ensure that the app handles this error situation.
    - **Steps:**
      1. Log in to the admin account.
      2. Navigate to the "Import games" section.
      3. Select a file with the incorrect game type.
    - **Expected results:**
      - The app displays an error message indicating that the selected file is invalid.
    - **Actual results:**
      - The error message is displayed correctly.

- **Download game file template**
  - **Description:** Test the ability to download the game file template onto the device.
    - **Objective:** Ensure that the template file can be successfully downloaded to the current device without any issue.
    - **Steps:**
      1. Log in to the admin account.
      2. Navigate to the "Download game file template" section.
      3. Wait for the template file export process to complete.
      4. Access the Android device's file system.
      5. Navigate to the Android directory.
      6. Navigate to the app file folder and confirm that a text file has been downloaded.
    - **Expected results:**
      - The app displays a message indicating that the template is exported.
      - The file can be found in the corresponding folder.
      - The file content is correct.
    - **Actual results:**
      - The message is showing correctly.
      - The text file is download correctly into the corresponding folder.

· The downloaded file is correct.

- **View, edit, delete, and create game**
  - **Description:** Test the ability to delete, edit, create, and view games.
    - **Objective:** Ensure that all these functionalities are functional.
    - **Steps:**
      1. Log in to the admin account.
      2. Navigate to the "Game List" section.
      3. Verify that all the games in the database are shown correctly with their associated cards.
      4. Select a game to edit by clicking the edit icon next to the game.
      5. Update the selected game's data.
      6. Return to the "Game list" section and verify that the changes have been applied.
      7. Select a game to delete by swiping the game.
      8. Verify that the game is deleted from the database and that it no longer appears in the game list.
      9. Navigate to the 'Create game' section.
      10. Create a new game with correct data.
      11. Return to the "Game list" and verify that the new game is created and displayed correctly.
    - **Expected results:**
      · All functionalities (viewing, editing, deleting, and creating games) work without any issues.
    - **Actual results:**
      · The functionality for viewing, editing, deleting, and creating games is working as expected.

  - **Description:** Test error handling when deleting, creating, or editing a game.
    - **Objective:** Verify that all related errors are properly handled.
    - **Steps:**
      1. Log in to the admin account.
      2. Navigate to the "Game list" section.
      3. Select a game to edit by clicking the edit icon of each game.
      4. Update the selected game with incorrect information (e.g., duplicate game name, insufficient cards, leaving the game name or category blank).
      5. Verify that the game presents the corresponding error message.
      6. Return to the "Game list" section and verifies that the changes were not applied.
      7. Try to delete the only game available.

8. Verify that an error message appears, indicating that it is the last game and cannot be deleted.
9. Verify that the game is not deleted.
10. Navigate to the 'Create game' section.
11. Create a game with incorrect input (e.g., blank game name, blank category, insufficient cards).
12. Verify that the corresponding error messages are displayed.
13. Returns to the "Game list" section and verify that the game was not created.

* **Expected results:**
  · All error handling works correctly, preventing invalid edits, creations, or deletions, and appropriate error messages are displayed.
* **Actual results:**
  · Error handling functionality for games works as expected.

- **View, delete, and create cards**
  - **Description:** Test the functionality of viewing, deleting, and creating cards.
    * **Objective:** Ensure that all these functionalities are functional.
    * **Steps:**
      1. Log in to the admin account.
      2. Navigate to the "Create cards" section.
      3. Verify that all non-panel cards are showing correctly.
      4. Long press the selected card and confirm that the delete icon appears.
      5. Delete the selected card.
      6. Verify that both the selected card and its matching pair are deleted.
      7. Go to the game list and verify that games containing the deleted card are updated correctly. If a game no longer meets the requirement of having 6 cards due to the deletion, it is automatically removed from the game list.
      8. Create a matching pair of cards.
      9. Verify that the newly created cards are created correctly.
    * **Expected results:**
      · All functionalities work without any issues, and the app updates games and cards properly.
    * **Actual results:**
      · All the functionalities work as expected.
  - **Description:** Test the app's ability to handle different card-related exceptions.
    * **Objective:** Ensure that errors are handled correctly and error messages are displayed.
    * **Steps:**

1. Log in to the admin account.
2. Navigate to the "Create cards" section.
3. Long press on a selected card and verify that the delete icon appears.
4. Try to delete the card associated with the last game we have.
5. Confirm that an error message is displayed and the card cannot be deleted.
6. Try to create a matching pair with errors (e.g., blank name, no card type selected, name already exists).
7. Verify that the card is not created and the corresponding error message is shown.

* **Expected results:**
  - Error messages are displayed correctly, and invalid operations are prevented.
* **Actual results:**
  - All the functionalities work as expected.

- **View, delete and create the panel card**
  - **Description:** Test the ability to delete, create, and view panel cards.
    * **Objective:** Ensure that all these functionalities are functional.
    * **Steps:**
      1. Log in to the admin account.
      2. Navigate to the "Create panel cards" section.
      3. Verify that all panel cards are showing correctly.
      4. Long press on a selected card and check that the delete icon appears.
      5. Delete the selected panel card.
      6. Verify that the selected panel card is deleted.
      7. Create a new panel card.
      8. Verify that the new panel card is created.
    * **Expected results:**
      - All functionalities (create, delete, view) work correctly without any issue.
    * **Actual results:**
      - All functionalities work correctly as expected.

- **Play the game**
  - **Description:** Tests the functionality of playing the provided game.
    * **Objective:** Verify that the game can be played without issues.
    * **Steps:**
      1. Log in to the user account.
      2. Select a video to play.

3. Wait for the timer to finish.
4. Play the game by dragging and dropping matching card pairs.

* **Expected results:**
  - The matched cards disappear from the screen.
  - Once the game is completed, the user is returned to the video screen or moved to the next game.
  - All associated audio is played correctly.
* **Actual results:**
  - The game runs smoothly without any problem.
  - All expected actions, such as audio playback and card disappearance, work as expected.

References

[1] "Normativa del treball final de grau del grau en enginyeria informàtica de la facultat d'informàtica de barcelona." Accessed: Sep. 24, 2024. [Online]. Available: https://www.fib.upc.edu/sites/fib/files/documents/estudis/normativa-tfg-gei-final.pdf.

[2] AppMySite. "What are aab files?" Accessed: Oct. 14, 2024. [Online]. Available: https://www.appmysite.com/blog/what-are-aab-files/.

[3] Wikipedia. "Picture communication symbols." Accessed: Sep. 24, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Picture_communication_symbols.

[4] C. Matters. "What is aac? – communication matters." Accessed: Sep. 24, 2024. [Online]. Available: https://www.communicationmatters.org.uk/what-is-aac/.

[5] T. D. US. "Picture communication symbols (pcs)." Accessed: Sep. 26, 2024. [Online]. Available: https://us.tobiidynavox.com/products/picture-communication-symbols-pcs.

[6] AssistiveWare. "Proloquo2go - aac app with symbols - assistiveware." Accessed: Sep. 24, 2024. [Online]. Available: https://www.assistiveware.com/products/proloquo2go.

[7] "Coughdrop - every voice should be heard." Accessed: Sep. 24, 2024. [Online]. Available: https://www.mycoughdrop.com/.

[8] "Autism ihelp – play." Accessed: Sep. 24, 2024. [Online]. Available: https://apps.apple.com/us/app/autism-ihelp-play/id521485216.

[9] J. Martins. "What is kanban? a beginner's guide for agile teams [2023]." Accessed: Sep. 25, 2024. [Online]. Available: https://asana.com/resources/what-is-kanban.

[10] S. Bhaskar. "What is scrum methodology? & scrum project management." Accessed: Sep. 25, 2024. [Online]. Available: https://www.nimblework.com/agile/scrum-methodology/.

[11] Wikipedia Contributors. "Github." Accessed: Sep. 25, 2024. [Online]. Available: https://en.wikipedia.org/wiki/GitHub.

[12] Wikipedia. "Taiga (project management)." Accessed: Sep. 25, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Taiga_(project_management).

[13] Overleaf. "Overleaf, online latex editor." Accessed: Oct. 01, 2024. [Online]. Available: https://www.overleaf.com/.

[14] GanttProject. "Ganttproject." Accessed: Oct. 01, 2024. [Online]. Available: https://www.ganttproject.biz/.

[15] "Visual studio code." Accessed: Oct. 01, 2024. [Online]. Available: https://code.visualstudio.com/.

[16] "Android studio - viquipèdia, l'enciclopèdia lliure." Accessed: Oct. 01, 2024. [Online]. Available: https://ca.wikipedia.org/wiki/Android_Studio.

[17] Draw.io. "Flowchart maker & online diagram software." Accessed: Oct. 01, 2024. [Online]. Available: https://app.diagrams.net.

[18] Figma. "Figma: The collaborative interface design tool." Accessed: Oct. 01, 2024. [Online]. Available: https://www.figma.com/.

[19] A. Developers. "Google play console | android developers." Accessed: Oct. 01, 2024. [Online]. Available: https://developer.android.com/distribute/console.

[20] Glassdoor. "Company salaries." Accessed: Oct. 07, 2024. [Online]. Available: https://www.glassdoor.es/Sueldos/index.htm?countryRedirect=true.

[21] G. Support. "How to use the play console - play console help." Accessed: Oct. 08, 2024. [Online]. Available: https://support.google.com/googleplay/android-developer/answer/6112435?hl=en.

[22] Atlassian, *Gitflow workflow*, (accessed Nov. 28, 2024), 2019. [Online]. Available: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow.

[23] General Data Protection Regulation (GDPR), *Art. 5 GDPR – Principles relating to processing of personal data | General Data Protection Regulation (GDPR)*, (accessed Nov. 08, 2024), 2018. [Online]. Available: https://gdpr-info.eu/art-5-gdpr/.

[24] Intersoft Consulting, *General data protection regulation (GDPR)*, (accessed Nov. 08, 2024), 2016. [Online]. Available: https://gdpr-info.eu/.