Th Islamic University of Gaza
Faculty of IT

الجامعة الإسلامية بغزة
كلية تكنولوجيا المعلومات

An Interactive Android Application for Gamified Early Childhood Learning

تطبيق لنظام الأندرويد تفاعلي لتعليم الأطفال بأسلوب اللعب

By

Doaa Emad Eldalu	220201524
Esraa Mazen Tabash	220207029
Malak Derar Abu Sisi	220193078
Noura Raed Abu Sharia	220191103

Supervised by

Ahmed Nasser Mahmoud
Master of Electrical and
Computer Engineering at
the University of Ottawa, Canada

Nada Usama Fathy Attia
Master of Electrical and
Computer Engineering at
the University of Ottawa, Canada

A graduation project report submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Technology

July-2025

Abstract

The increasing use of mobile devices by children, particularly those aged 4 to 10, has resulted in excessive exposure to non-educational content, such as social media and entertainment-focused applications. This situation raises concerns about cognitive development and the potential risk of encountering inappropriate material.

To address this issue, an educational mobile application for Android was developed targeting children aged 4 to 10. The app provides engaging and interactive games, designed to enhance memory, attention, and thinking skills within a safe and age-appropriate environment.

The development process began with an analysis of how children in the target age group engage with digital content and educational materials. Based on this analysis, intuitive and age-appropriate interface designs were planned. These designs were then transformed into interactive digital prototypes prioritizing ease of navigation, visual appeal, and educational value. The app was developed using Kotlin and XML for the user interface, while Firebase was utilized for data storage and user authentication. Additionally, the educational games integrated into the app were developed using the Unity game engine, with scripts written in C# to ensure smooth performance and interactive gameplay.

Following development, the app was tested by a group of users. Feedback indicated that the app is user-friendly, enjoyable for children, and effective in supporting learning through visuals and interactive activities.

This project demonstrates that a well-designed educational application can positively support children's cognitive and emotional development, particularly in areas affected by conflict where access to traditional education may be limited. Future improvements could include adding multilingual support and integration with schools and educators to expand reach and impact.

ملخص الدراسة

يهدف المشروع إلى تطوير تطبيق تعليمي تفاعلي موجه للأطفال في الفئة العمرية من 4 إلى 10 سنوات، وذلك نظراً للانتشار الواسع لاستخدام الأجهزة الذكية بين الأطفال وما يترتب عليه من تعرض لمحتوى غير تعليمي أو غير مناسب للأعمار. لوحظ أن العديد من التطبيقات المتوفرة تركز على الترفيه فقط دون مراعاة الجانب التعليمي أو التربوي.

تم تصميم وتطوير تطبيق يعمل على نظام أندرويد، يحتوي على مجموعة من الألعاب التعليمية الممتعة التي تهدف إلى تنمية مهارات التفكير والتركيز والذاكرة ضمن بيئة آمنة ومناسبة للأعمار. تم بناء واجهات المستخدم باستخدام لغة Kotlin وXML، في حين استخدم Firebase لإدارة بيانات المستخدمين وتخزين المعلومات. كما تم تطوير الألعاب التعليمية المدمجة باستخدام محرك الألعاب Unity ، مع برمجتها بلغة C# لضمان تفاعل سلس وأداء جيد.

شملت مراحل العمل دراسة خصائص تعلم الأطفال وآليات جذب الانتباه، ثم تصميم واجهات مرئية بسيطة وتفاعلية. تم تقييم التطبيق من خلال تجربة مجموعة من المستخدمين، حيث أظهر التطبيق سهولة في الاستخدام وفوائد تعليمية واضحة.

توصلت الدراسة إلى أن التطبيقات التعليمية المصممة بعناية تساهم في دعم نمو الأطفال المعرفي والعاطفي، خصوصاً في المناطق التي تفتقر إلى التعليم التقليدي. تُقترح في المستقبل توسعة نطاق التطبيق ليشمل لغات متعددة، وربطه مع المدارس والمعلمين لزيادة الأثر والوصول إلى عدد أكبر من الأطفال.

Dedication

This project is dedicated to:

Our professors and mentors: Heartfelt thanks are extended to all professors and mentors for their invaluable guidance, knowledge, and support throughout this journey. Their passion for teaching and dedication to nurturing our skills have profoundly shaped our understanding of information technology and inspired us to achieve our goals.

Our families: Deepest gratitude goes to our families for their unwavering encouragement, patience, and love. Their sacrifices and belief in our abilities have been the foundation of our success. This achievement stands as a tribute to their constant support and the values they have instilled in us.

Our team members: Special appreciation is given to our fellow team members for their commitment, cooperation, and shared vision. The diverse skills and perspectives each member brought to the project were essential in making this work possible. The camaraderie and collaboration made this journey both productive and enjoyable.

To everyone who supported us along the way, sincere thanks are offered. Your confidence and encouragement have been vital to our academic and personal growth.

And to every child in Gaza and all around the world who deserves a safe space to grow, learn, and shine — this work is for you.

This work is dedicated to all who contributed to our IT graduation journey. Your guidance and inspiration played a significant role in shaping our achievements, and for that, we are profoundly grateful.

Acknowledgment

As a team, we extend our sincere gratitude and appreciation to all individuals and organizations who supported and contributed to the successful completion of this project.

Special thanks to our supervisor for their invaluable guidance, mentorship, and expertise throughout every stage. Your continuous encouragement, insightful feedback, and unwavering dedication played a key role in shaping the quality and direction of our work.

We are also deeply thankful to our professors and mentors, whose knowledge, support, and passion for teaching have inspired us and strengthened our academic journey.

To the external experts who generously shared their knowledge and insights, we are grateful for your valuable contributions that enhanced the development of our project and its educational impact.

Our appreciation extends to our university for providing the necessary tools, resources, and environment that facilitated the progress of our work.

To our families and friends — your love, patience, and unwavering belief in us have been a constant source of strength and motivation. We are forever grateful for your sacrifices and encouragement.

And to every child in Gaza and all around the world who deserves a safe space to grow, learn, and shine — this work is for you.

Finally, we acknowledge every individual who participated in testing, provided feedback, or supported us directly or indirectly throughout this journey. Your contributions are deeply appreciated and have left a lasting impact on the success of this project.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Full Form
APK	Android Package
AR	Augmented Reality
DB	Database
IDE	Integrated Development Environment
OOP	Object-Oriented Programming
OS	Operating System
SDLC	Software Development Life Cycle
UI	User Interface
UX	User Experience
XML	eXtensible Markup Language

Chapter 1

Introduction

Chapter 1
Introduction

In recent years, the rapid growth in the use of digital devices such as smartphones and tablets has significantly increased the demand for applications and websites tailored to the diverse needs of various communities. This surge has created a greater need for developers skilled not only in programming but also in designing intuitive and user-friendly interfaces, as poor design often leads to app abandonment and reduced user satisfaction [1].

At the same time, the number of children using digital devices has risen considerably. Many young children are able to operate tablets and smartphones early in life, which has important implications for learning and development. Surveys show that a majority of children and parents use at least one digital device daily, with smartphones and tablets being the most common [2].

This increased exposure has spurred the growth of interactive educational applications that combine play and learning. These apps—featuring puzzles, coloring, and problem-solving games—have proven effective in enhancing cognitive and language skills, making digital learning tools integral to early childhood education and cultural engagement [3].

1.1 Problem Statement

The rapid development and widespread use of mobile applications have contributed to the increasing popularity of social media platforms, primarily due to their ease of access, engaging features, and stimulation of visual and auditory senses. As a result, a growing number of children are being exposed to non-educational content through platforms such as YouTube and TikTok. This exposure has raised serious concerns regarding cognitive and behavioral development, increasing vulnerability to inappropriate content.

In many regions with underdeveloped educational systems—such as areas affected by conflict, poverty, and limited infrastructure—access to quality learning resources remains a significant challenge. Therefore, there is a pressing need for digital platforms that prioritize educational development, foster cultural identity, and offer engaging content that supports the learning process in safe and interactive ways [4].

1.2 Objectives

1.2.1 Main Objective

Enhancement of cognitive and academic skills for children through the development of an engaging mobile application and a complementary website. The platform is designed to utilize children's time effectively by providing educational content in the form of interactive and entertaining games.

1.2.2 Sub Objectives

Specific objectives of the project include:

1. Improving abilities in reading, writing, and mathematics through playful and engaging learning experiences.
2. Designing interactive games that are both educational and entertaining, encouraging active participation.
3. Fostering a love for learning and stimulating curiosity among young learners.

4. Promoting the application among parents in conflict-affected regions as a supportive educational tool in challenging circumstances.

1.3 Scope and Limitations

1.3.1 Scope:

The educational application focuses primarily on Arabic language content, targeting children between the ages of 3 to 10 years. It includes interactive games to teach letters, words, basic arithmetic, and creative activities such as coloring and painting.

1. Initial release will target the Android platform via the Google Play Store, with expansion to iOS planned in future development stages.
2. The project is expected to be completed in over three months, aiming to deliver a fully developed, ready-to-use educational tool rather than a basic prototype.

1.3.2 Limitations:

1. Content is limited to children aged 3 to 10 years.
2. Initial availability restricted to Android devices only.
3. Educational content will be presented in Arabic exclusively in the first version, limiting accessibility for non-Arabic speakers.

1.4 Importance of the project

This project addresses the need to redirect children's screen time toward learning and the development of essential skills. The application promotes skill enhancement through interactive games and engaging stories while monitoring engagement with educational content. Activities such as puzzle-solving, coloring, and foundational learning contribute to overall development.

Benefits of Using Interactive Applications in Learning:

1. Active Participation: Encourages learners to engage directly with content, promoting deeper understanding.
2. Real-Life Experiences: Simulated scenarios aid in comprehension and application of learned concepts.
3. Easy Access to Education: Educational content is accessible globally, ensuring continuous learning.
4. Improved Skills and Abilities: Supports development of cognitive, linguistic, and problem-solving skills.
5. Support for Early Learning: Enhances reading skills and general knowledge, especially for early or underperforming learners.

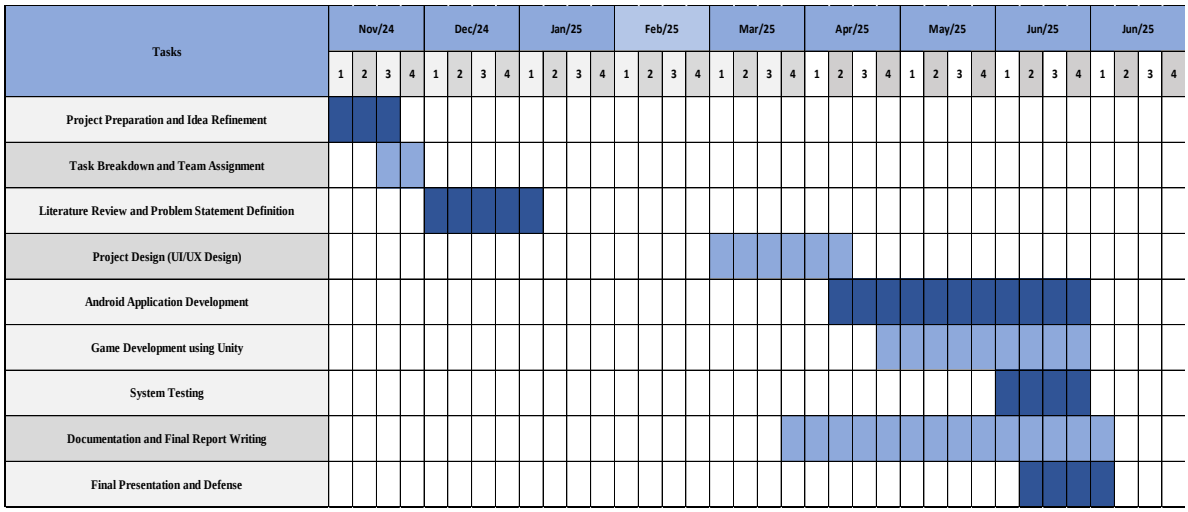


Figure 1.4-1 Time Table

Chapter 2
Background

Chapter 2
Background

2.1 Digital Transformation in Educational App Development for Children

The use of mobile applications in children's education has grown significantly in recent years, especially among children aged 4 to 10. These applications aim to enhance cognitive skills such as thinking, memory, and concentration by combining learning with fun and entertainment.

Despite their popularity, many of these apps lack a well-structured and thoughtful design, which limits their educational effectiveness and leads to low user engagement.

Traditional educational methods have become less appealing to today’s tech-savvy children, who prefer interactive tools that align with their interests and offer an enjoyable learning experience.

Therefore, educational applications present a valuable opportunity to transform the way children learn, provided they effectively integrate quality educational content with engaging and age-appropriate entertainment elements.

In this chapter, these tools have been studied and analyzed to provide a more engaging and interactive educational experience for children.

2.1.1 Importance of Educational Apps for Children

Educational apps designed for children aged 4–10 play a vital role in early childhood development. They offer a safe and interactive environment where children can:

- Develop cognitive and motor skills through interactive games.
- Learn foundational topics such as numbers, letters, colors, and shapes.
- Improve focus, memory, and problem-solving abilities through well-designed challenges.

- Engage in independent learning at their own pace.

By combining gamification, storytelling, and personalized feedback, educational apps help children retain information better while staying entertained.

2.1.2 Objectives of the Educational App

The main objective of the project is to develop a mobile application that enhances children's thinking, memory, and concentration skills through fun and interactive games. The specific goals include:

• Providing age-appropriate educational content.
• Designing an intuitive and engaging user interface for children.
• Including a reward-based system to motivate continued learning.
• Ensuring a safe, ad-free, and child-friendly environment.
• Offering parental controls and progress tracking features.

2.1.3 What Factors Motivate Children to Engage with Mobile Apps?

Children's engagement with mobile apps is influenced by several psychological, technical, and educational factors. Here are the most prominent ones:

1. Attractive and Visual Design:
   • Bright colors and animations.
   • Fun and likable cartoon characters.
   • A simple and easy-to-understand interface.

2. High Interactivity:
   • Immediate response to touch.
   • Sounds and encouraging feedback.
   • Mini-games and rewards (gamification).

3. Appropriate Challenge Level:
   • Tasks suitable for the child's age and skill level.
   • Different levels of difficulty.

4. Balanced Educational and Entertainment Content:
   • Integrating learning with play.
   • Activities that develop skills like reading, counting, and logical thinking.

5. Repetition and Reinforcement:
   • Repeating lessons in engaging ways.
   • Positive reinforcement through stars or virtual prizes.

6. Storytelling Elements:
   • Apps with a motivating storyline increase engagement.
   • Children follow characters and plot with curiosity.

7. Parental Involvement:
   • Features that allow parents to track progress.
   • Guidance on how to support learning at home.

8. Appropriate Audio:
   • Child-friendly voices and clear pronunciation.
   • Sound effects that enhance understanding.

9. Freedom of Choice:
   • Allowing children to choose activities boosts independence and motivation.

2.2 Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is a structured process used to design, develop, and maintain high-quality software in an efficient and cost-effective way.

2.2.1 Phases of SDLC (Adapted for the Educational App Project)

1. **Planning:**

   This is the initial phase where the project's goals, scope, budget, and timeline are defined. It's also where the feasibility of the project is assessed.

2. **RequirementsAnalysis:**

   Developers and stakeholders gather and analyze the functional and non-functional requirements of the software. This ensures a clear understanding of what the software must do.

3. **Design:**

   In this phase, the software's architecture and design are created. This includes database design, user interface layout, and system flow.

4. **Implementation(Development):**

   Developers write the actual code based on the design specifications using the chosen programming languages and tools.

5. **Testing:**

   The software is tested to find and fix bugs or issues. This ensures the system works correctly and meets the requirements.

6. **Deployment:**

   After testing, the software is released for use. This could be a limited release or a full rollout, depending on the project.

7. **Maintenance:**

   Once deployed, the software is monitored and updated as needed. This includes fixing bugs, adding new features, or improving performance.

Why SDLC is Important:

- It provides a clear roadmap for the development team.
- Helps manage time, cost, and resources effectively.
- Reduces the risk of project failure.
- Ensure the final product meets the user's expectations and quality standards.

Chapter 3

Related Works

Chapter 3
Related Works

3.1 Introduction

This chapter discusses several previous studies and projects that are related to educational and entertainment mobile applications for children. The focus is on the design, interaction strategies, challenges, and learning goals. Each reviewed project shares similarities with the current work in some aspects, while also offering points of contrast. At the end of the chapter, a detailed comparison highlights the similarities and differences with the proposed system.

Anjar Sari et al. (2024) developed a mobile learning game prototype called *Monster Pizza* to teach fractional concepts to third-grade elementary students. The interactive scenario involves dividing pizzas based on fraction concepts, including identification, comparison, ordering, and operations with like denominators. Following a Research and Development approach, the game was iteratively designed and validated with 35 students. It received high content feasibility and presentation design scores, with recommendations for further enhancements such as voice-over instructions and clearer fonts to expand the game's scope [5].

Ahmad et al. (2024) examined a web-based educational application for children aged 5 to 7 developed using the ADDIE instructional design model. The application includes multimedia content, interactive exercises, and a user-friendly interface aimed at enhancing early reading, writing, and numeracy skills. User feedback indicated improved motivation, focus, and learning outcomes, with 92% of participants achieving good or excellent grades. The study underscores the value of pedagogically grounded digital tools in supporting early childhood education [6].

Kotserkova, Littleton, and Fluet (2022) explored how children aged 2 to 8 interact with digital technologies at home, emphasizing parental involvement, child autonomy, and digital literacy. Using qualitative data from British families, the study highlights children's active role in discussing digital experiences and parents' role as mediators. The findings indicate that highly rated educational apps have greater potential despite widespread language quality issues, with recommendations for practical guidance to parents and educators on app selection and policymaker advice for improved rating systems and regulation [7].

In response to the COVID-19 pandemic, Sureshwaran et al. (2022) proposed a digital learning platform leveraging modern technologies such as Text-to-Speech (TTS), Machine Learning (ML), Image Processing, and Web Scraping to automatically generate educational videos for children. Starting with keywords, the system retrieves relevant content, simplifies language, produces voice narration, and combines it with suitable images. Natural Language Processing (NLP) is used to summarize and tailor content to children's comprehension levels. The study highlights visual learning advantages over traditional methods and addresses challenges of manual video production by automating the process, resulting in improved engagement, comprehension, and retention, as well as reduced production time and cost [8].

Aydoğdu (2021) employed a quasi-experimental design with 26 preschool children (aged 4–5) to assess augmented reality (AR) in educational content delivery. The experimental group received AR-supported instruction while the control group used traditional methods. Pre- and post-tests measured motivation, attention, and conceptual understanding, revealing statistically significant gains in all three dimensions for the AR group compared to controls. A complementary study using the ABCD-AR application further showed increases in mean scores and reduced time to complete tasks, indicating improved competence and efficiency. The results demonstrate AR's potential to engage preschoolers, enhance focus, and accelerate conceptual learning [9].

Manzano-León et al. (2021) conducted a comprehensive systematic literature review on the effects of gamification in education, analyzing 14 experimental or quasi-experimental studies published between 2016 and 2020 involving 5,071 students across various educational levels and environments. The findings revealed a large overall effect size of 0.822 (Hedges' g), indicating a statistically significant and strong positive impact of gamification on educational outcomes. Gamification was shown to improve student motivation, engagement, and academic performance, especially when incorporating game elements such as points, badges, levels, and leaderboards. The review also found that shorter interventions (less than one month) and digital learning environments tend to be more responsive to gamified strategies compared to traditional settings. The authors caution against overusing gamification without clear pedagogical alignment and recommend further research focusing on personalization, age-appropriate design, and subject-specific adaptations to maximize long-term impact [10].

In related work, Meryl Alper and colleagues (2012) explored interactive technologies for children with special needs, focusing on participatory design, assistive tools for the hearing impaired, and tangible computing. The paper emphasizes involving children with disabilities, their families, and caregivers throughout the design process to ensure technologies are appropriately tailored. One experimental case demonstrated that deaf children who used 3D virtual reality environments for enhancing inferential reasoning outperformed those using 2D interfaces by 40%. Another case study from India applied tangible sensory tools with children with autism, resulting in an estimated 60% improvement in social interaction and joint attention during the trial period. The study highlights key principles for effective design: deep user engagement, interdisciplinary collaboration, individualized approaches, and practical implementation to ensure sustained impact beyond labs or formal classrooms [11].

Finally, Myrat Durdyyev (2012), in his final-year project at Universiti Teknologi PETRONAS, developed a mobile learning application titled *Fun Math* targeting primary school children. The study aimed to transform mathematics from a subject perceived as boring and difficult into an engaging, interactive experience. Using Google App Inventor, the app integrates inductive and deductive teaching methods alongside visual aids like fruits and animals, and includes features such as quizzes, customizable tests, tutorial videos, and progress tracking. The methodology followed a structured software development process, beginning with requirement gathering through observation and literature review, followed by interface design using block-based coding, implementation with Android emulator testing, and evaluation through usability testing involving real users (children). User testing with 10 children showed that 90% found the app functional, 100% liked its interface and ease of use, and 70% expressed a desire to continue using it. The study concludes that mobile applications like *Fun Math* can effectively support early mathematics education when designed with usability and child engagement in mind [12].

Table 3.1.1 Related Studies Comparison

Researcher	Similarities with Current Project	Differences and Unique Features of the Study
Anjar Sari et al. (2024)	- Developed an interactive educational game. - Teaches math concepts through gameplay.	- Monster Pizza game for teaching fractions. - Received high content and design scores with recommendations for improvement.
Ahmad et al. (2024)	- Web-based educational app for children. - Focus on early literacy and numeracy skills.	- Uses the ADDIE instructional design model. - Evaluates student motivation and performance improvements.
Kotserkova, Littleton, Fluet (2022)	- Studies children's interaction with digital apps. - Highlights parental role in digital learning.	- Qualitative study of British families. - Focus on language quality and app rating systems.
Sureshwaran et al. (2022)	- Uses multimedia to enhance learning. - Improves children's engagement and comprehension.	- AI-driven system for auto-generating educational videos. - Reduces production cost and time.
Aydoğdu (2021)	- Supports learning with modern technologies. - Improves children's motivation and attention.	- Augmented reality study with preschool children. - Detailed pre- and post -Intervention measurements.
Manzano-León et al. (2021)	- Focus on gamification to motivate children. - Interactive digital learning environments.	- Systematic literature review. - Emphasizes pedagogical alignment and cautions against overusing gamification.

Researcher	Similarities with Current Project	Differences and Unique Features of the Study
Meryl Alper et al. (2012)	- Designed for children with special needs. - Involves families and caregivers in the design process.	- Uses 3D virtual reality and tangible sensory tools. - Significant improvements in social interaction and focus.
Myrat Durdyyev (2012)	- Educational app for primary school children. - Use of visual and motivational aids. - Includes quizzes, videos, and progress tracking.	- Developed using Google App Inventor. - Features more detailed quizzes and tutorial videos.

Chapter 4

Methodology

Chapter 4
Methodology

4.1 Introduction

This chapter presents the methodology adopted in the development of the "Majarat Al-Ma'rifa" application. The Software Development Life Cycle (SDLC) was chosen as the primary framework to organize the development process—from the planning phase to the final implementation—due to its clear structure that ensures quality and helps achieve goals in a systematic and well-planned manner.

The chapter discusses the rationale behind choosing this methodology, along with a detailed overview of its phases, emphasizing the characteristics that made it suitable for the nature of the project and the targeted educational environment. The methodology aligned well with the project's defined requirements, contributing to efficient and organized execution.

4.2 Software Development Methodology

4.2.1 Name of the Methodology

Adopted the Software Development Life Cycle (SDLC) methodology.

4.2.2 Rationale for Choosing the Methodology

SDLC was chosen due to its structured and systematic approach, which helps manage the project in clear, sequential phases from planning to maintenance. It is especially suitable for academic projects where requirements are known in advance.

4.2.3 Key Practices and Phases

The main steps of the methodology as applied in the project were:

1. **Planning**

The main objective of the application is to enhance thinking, memory, and concentration skills for children aged 4–10 in a fun, interactive, and safe environment. During this stage, the project team identified resources, outlined the budget, and defined the project timeline.

2. **Requirement Analysis:**

A detailed analysis of the needs of key stakeholders (children, parents, and educators) was conducted. Core features identified include:

- Fun games and puzzles such as image completion.
- Letter-specific challenges.
- Matching letters with words that start with them.
- Alphabet learning videos that children can repeat aloud (Future plan).
- Dotted letters and shapes that children can trace and redraw.
- Matching numbers with the correct quantity of shapes.
- English alphabet exploration (Future plan).
- Animated characters that provide tips and educational messages.
- Rewards in the form of stickers and badges.
- Space exploration and discovery modules.

3. **System Design**

The application was designed using a modular approach—each game or activity is implemented as a separate module. This architecture allows for easy updates or the addition of new content without affecting the core system.

- The interface is child-friendly, with large buttons, colorful illustrations, and intuitive navigation.
- Firebase Authentication is used to securely manage child accounts, with account creation allowed only for users aged 6 and above.
- User data (e.g., name, profile picture, game progress) is securely stored in Firestore.
- The entire interface supports Arabic only, with plans to support additional languages in the future.
- System design also ensures compatibility with a wide range of Android devices, with optimized performance and minimal battery usage.

4. Implementation

The app was developed using modern cross-platform tools, enabling smooth performance across Android devices. Animations, sound effects, and multimedia content were integrated to increase engagement. All interfaces and content are fully localized in Arabic to cater to the target age group. Building the main interfaces using Kotlin and designing them with XML, developing interactive games using C# and the Unity engine, integrating Lottie animations and sound effects to attract children, using Firebase Authentication for secure accounts, and storing the child's data and progress in the cloud via Firestone.

5. Testing

Comprehensive testing was conducted, including:

- Unit testing for individual game modules.
- Usability testing to ensure the interface is age-appropriate and intuitive.
- Performance testing to guarantee smooth operation across devices and avoid excessive battery drain.
- Security testing to protect user data, especially children’s personal information.

6. Deployment

The app is currently deployed on Android platforms through the Google Play Store. A marketing strategy and app descriptions were tailored to communicate the app's educational value to parents and educators.

7. Maintenance

Ongoing maintenance includes bug fixing, performance improvements, and content updates. The modular design allows for easy integration of new features or games.

Future plans include adding language options and linking child accounts with parent dashboards for better monitoring.

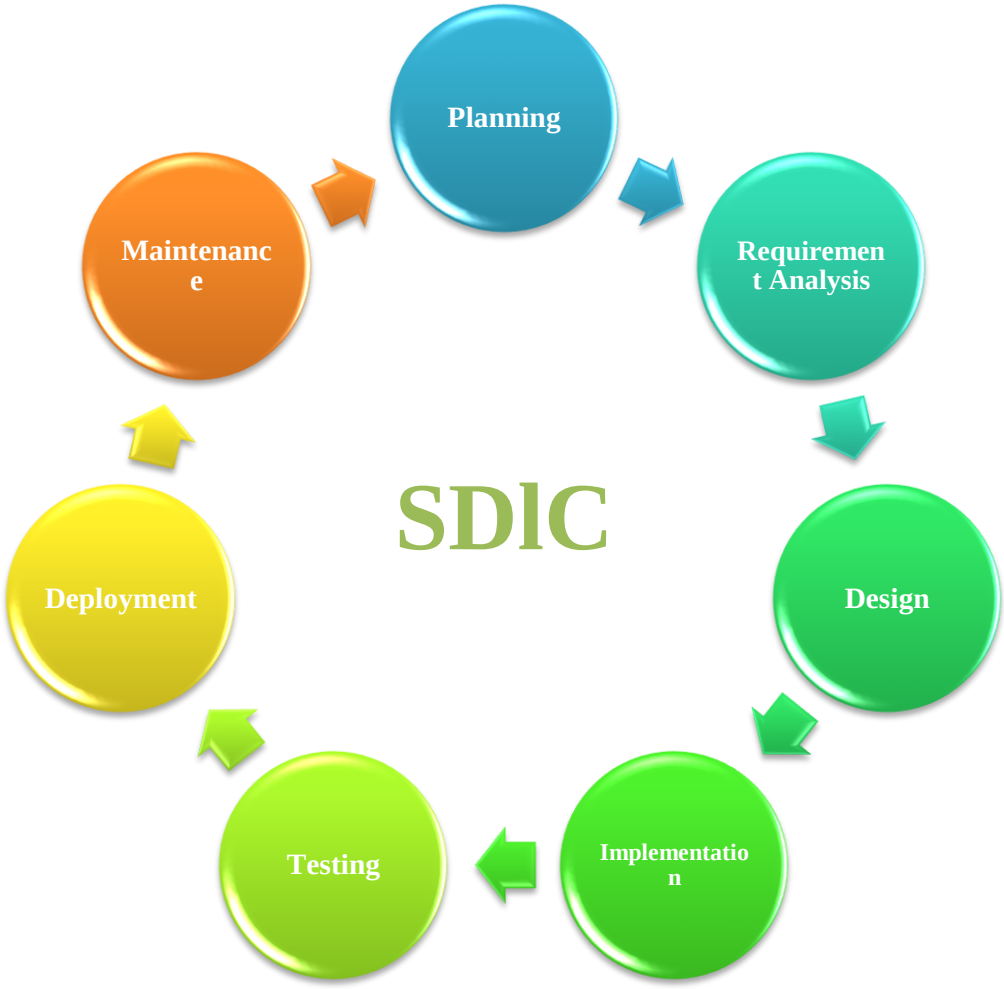


Figure 4.21- SDLC Cycle

4.3 Summary

This chapter presented the methodology followed in the development of the educational games application. A simplified Software Development Life Cycle (SDLC) approach was adopted, and the work was divided into three main phases: Preparation, Development, and Evaluation. This structure helped organize tasks and ensure steady progress. As a result, the project will be able to develop an interactive, engaging, and child-friendly application that meets children's educational needs innovatively and enjoyably.

Chapter 5

Requirements

Chapter 5
Requirements

5.1 Introduction

This chapter discusses the essential requirements that were relied upon in the development of the educational games application, including the software and hardware tools, programming languages and frameworks used, as well as the methodology followed to organize the project stages. The project was divided into three main phases: preparation, development, and evaluation, to ensure effective task organization and continuous progress. The chapter also highlights team management, communication tools, and conflict resolution strategies to ensure a smooth and efficient workflow. This chapter aims to document all the technical and organizational requirements that support the development process of the application.

5.2 Business Model

The Business Model below provides a visual representation of the project's structure, value proposition, and strategic approach. It identifies the key components of the application's development and delivery process, including the main partners, customer segments, resources, activities, cost structure, and revenue streams. This model was designed to guide the team in maintaining a clear focus on the target users (children aged 4–10, parents, and educators), ensuring the educational and commercial success of the app.

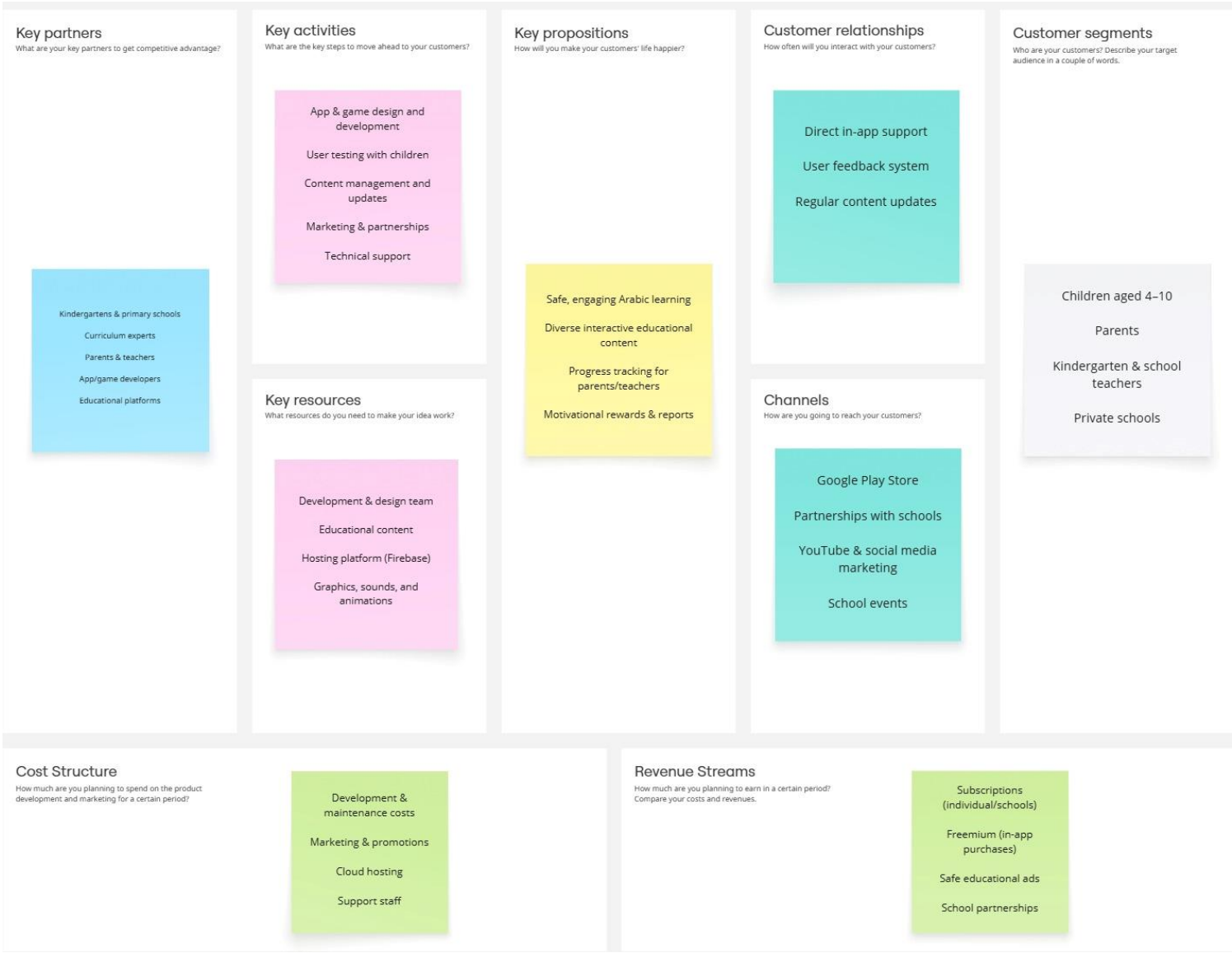


Figure 5.2-1 Business Model of the Educational Mobile App

5.3 Adaptations or Modifications to the Standard Methodology

The standard Scrum approach was simplified by omitting formal Scrum roles such as Scrum Master or Product Owner, due to the small team size. Instead of daily stand-up meetings, weekly planning and review sessions were used. These modifications maintained productivity and focus without overcomplicating the process.

5.4 Tools and Equipment

Table 0.1 Software Tools

Tool	**Purpose**
Android Studio	IDE used for coding, designing layouts, and testing the app.
GitHub	Version control and collaboration platform to track code changes.
Firebase Cloud Firestore	A NoSQL cloud database is used to store and sync real-time user data.
Firebase Authentication	Service for managing user login and registration securely.
Figma	Used to design the app interface and create interactive prototypes.
Adobe After Effects	Used to create animations integrated into the app using Lottie.
Google Forms	Used to collect user feedback and conduct surveys during testing.

Table 0.2 Languages and Frameworks

Language/Framework	**Purpose**
Kotlin	The main programming language is used for Android development.
C#	Used for developing games with the Unity engine.
XML	Used for designing the user interface layout.
Lottie Animations	Framework used for rendering lightweight, scalable animations.

Table 0.3 Hardware Tools

Device	Specifications
Laptop (Development)	AMD Ryzen 5 4500U, 2.38 GHz, 16 GB RAM, 64-bit OS
Mobile Device (Testing)	Redmi Note 10 Pro Max, 6 GB RAM, 128 GB Storage

5.5 Project Phases

The development process was divided into three main phases: Preparation, Development, and Evaluation. This structure enabled efficient task organization and ensured continuous progress through iteration and feedback.

Preparation Phase

This initial phase focused on research, planning, and project setup.

Table 0.1 Preparation phase Activities

Activity	Description
Search & Review Similar Works	Researched similar educational apps to understand best practices and identify feature gaps.
Start Project	Initialized the project using Android Studio and set up GitHub for collaboration.
Plan & Divide Work	Defined responsibilities: one member focused on design, the other on development. Tasks were organized into modules.

Development Phase

The core functionality of the app was implemented during this phase through weekly sprints.

Table 0.1 Development phase Activities

Activity	**Description**
UI Design	Created colorful and child-friendly interfaces using Figma and implemented them in XML.
Feature Implementation	Built main features such as registration, login, and educational activities using Kotlin.
Weekly Meeting with Supervisor	Regularly presented updates, discussed challenges, and received feedback.
Receive Feedback & Adjust	Refined features and UI based on supervisor feedback and initial user impressions.
Iterate & Improve	Improved overall functionality and experience in response to feedback.

Evaluation Phase

In this final phase, the app was tested, refined, and prepared for submission.

Table 0.2 Evaluation phase activities

Activity	**Description**
Functional Testing	All verified features worked as intended without major bugs.
Usability Testing	Tested with a sample of children to assess ease of use and engagement.
Bug Fixing & Optimization	Resolved issues, improved app performance, and responsiveness.
Final Submission	Ensured readiness for submission by reviewing all app content and documents.

Timeline

Table 0.3 Timeline

Phase	**Weeks**	**Tasks**
Preparation	Week 1	Define objectives, analyze similar apps, and set up tools.
UI/UX Design	Weeks 2–4	Design screens and create animations.
Backend Setup	Week 5	Configure Firebase services and database.
Development & Integration	Weeks 6–8	Implement features, integrate design, and code.
Testing & Improvements	Weeks 9	Test the app, fix bugs, and apply feedback.
Finalization	Week 10	Final review, optimization, and documentation.

5.6 Team Management

Table 0.1 Team Structure and Roles

Member	**Role and Responsibilities**
UI/UX Designer	Created visual designs using Figma, built animations using After Effects, and Lottie.
Developer	Developed application features using Kotlin and Firebase, and integrated the design assets.

Table 0.2 Communication and Collaboration Tools

Tool	**Purpose**
GitHub	For version control and synchronizing code.
WhatsApp/Telegram	For quick communication and task updates.
Google Drive	For storing shared documents, assets, and reports.
Google Meet	For virtual meetings with the supervisor and team discussions.

5.7 Conflict Resolution Strategies

When conflicts or disagreements arose, they were resolved through open discussion. If consensus could not be reached, guidance was sought from the supervisor to ensure progress was not hindered.

5.8 Summary

This chapter provided a comprehensive overview of the requirements on which the project was built, including tools, technologies, work methodology, and organization. By selecting appropriate tools and applying a simplified and flexible software development approach, the team was able to carry out tasks efficiently within a defined timeframe. Clear division of work and well-defined roles also contributed to effective collaboration among team members. This chapter serves as a key reference for understanding the foundational structure that supported the project's success and its progress toward achieving its educational goals.

Chapter 6

Design

Chapter 6
Design

6.1 Introduction

In this chapter, a clear and concise explanation of the complete design of the application is provided, primarily aimed at helping children learn through fun and play. The interface is planned to be colorful and attractive so that children enjoy using it and do not get bored.

A walkthrough of the app flow is presented—from the moment the child opens the app until different learning activities are completed. This includes the overall structure, the connection between screens, and how the design supports smooth and enjoyable navigation.

The chapter also explores how data is stored and organized in a simple, structured way, ensuring effective communication between all parts of the app. The goal is to offer children a smooth, enjoyable, and beneficial experience that supports both learning and fun.

6.2 Wireframes

A wireframe is a two-dimensional illustration of a page's interface that specifically focuses on space allocation and prioritization of content, functionalities available, and intended . This is the wireframe for our application:

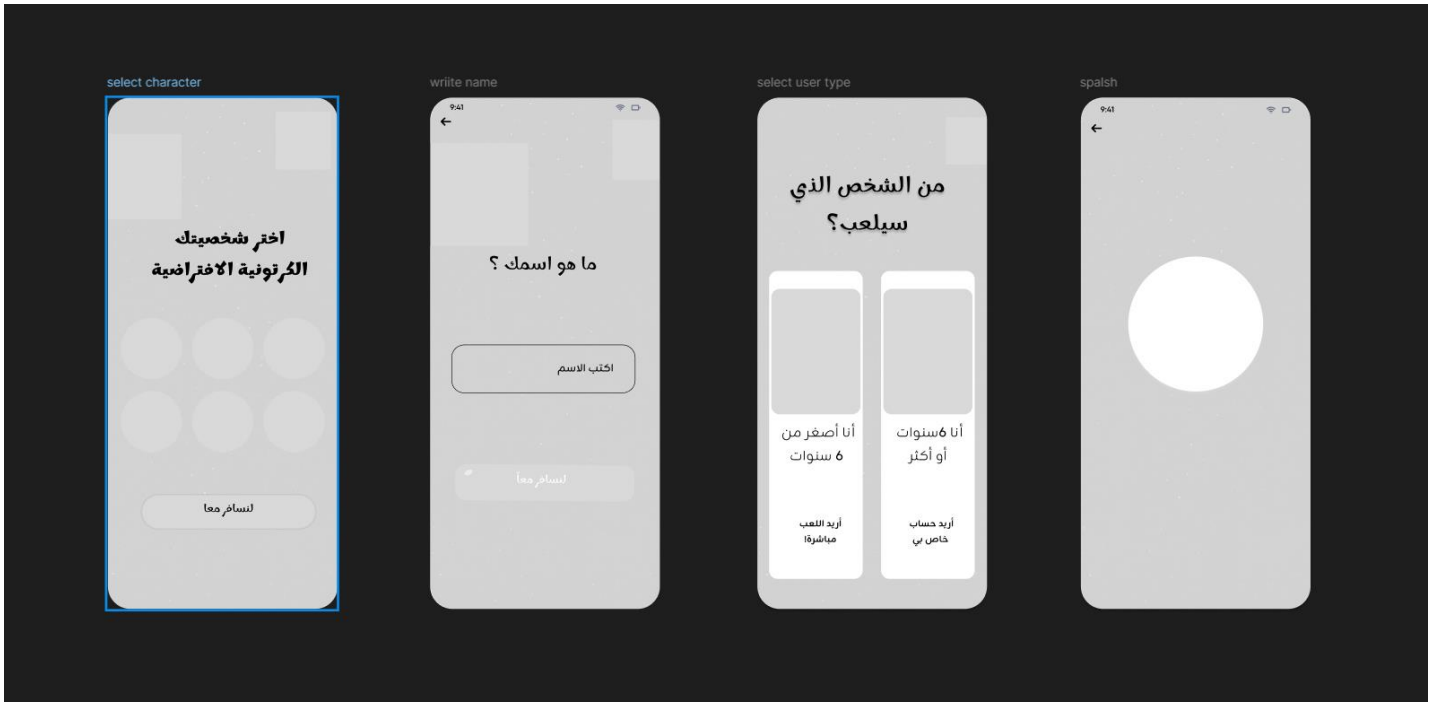


Figure 6.2-1 Wireframes 1

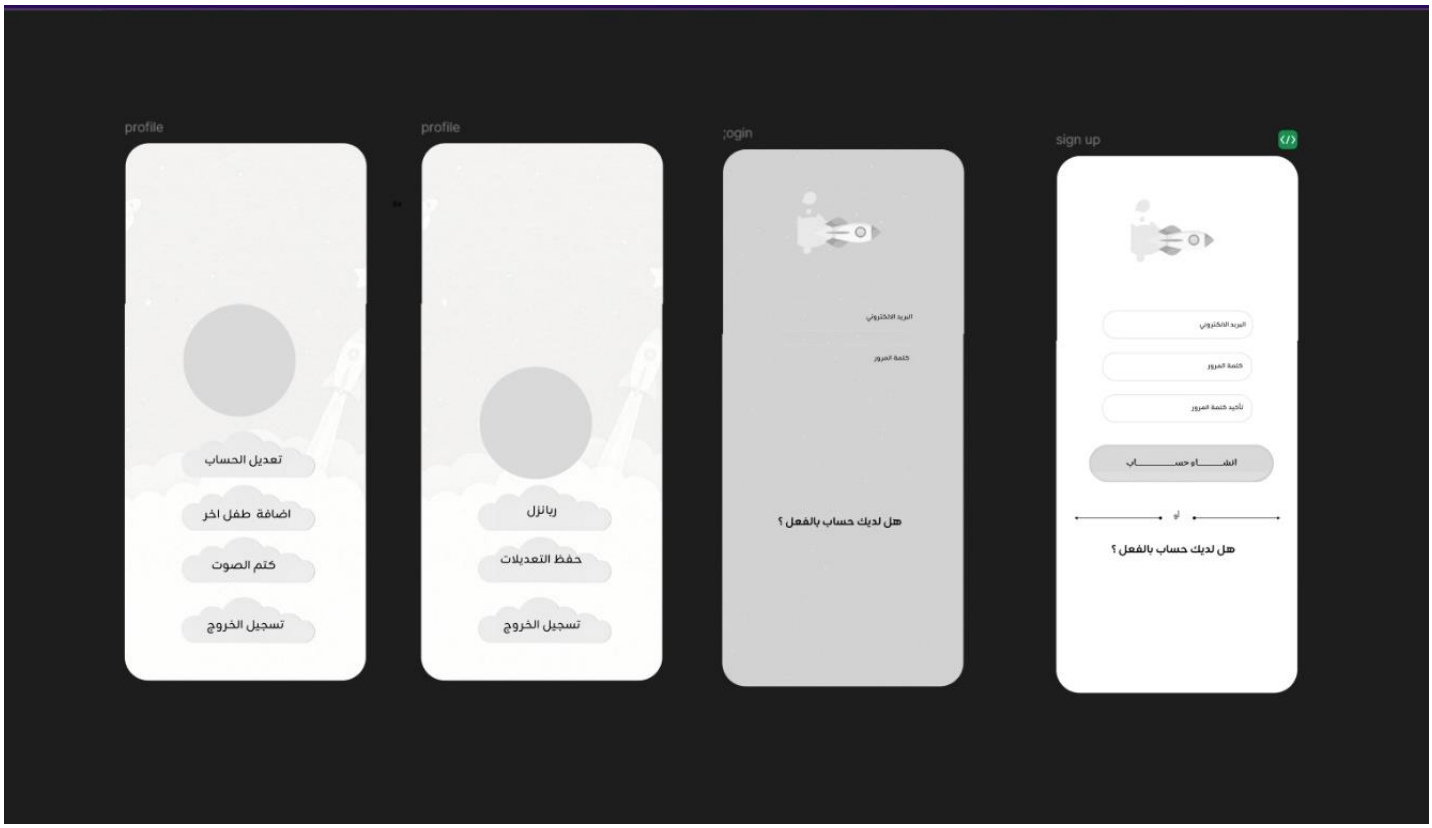


Figure 6.2-2 Wireframes 2

6.3 Application UI

- Splash Screen & Welcome Screens

In illustrates the Splash Screen and Welcome Screen, which serve as the entry point to the application. The splash screen displays a cheerful logo and a space-themed background with soft animations and a child's laughter sound effect. It aims to capture the child’s attention instantly and create a sense of joy and curiosity. After a brief loading period (7 seconds), the app transitions automatically to the welcome screen, which maintains the magical space atmosphere and presents a welcoming message before navigating to the user selection screen without requiring any interaction.

Figure 6.3-1 Splash Screen & Welcome Screens

- Create Account Screen & SignIn Screen & SelectUserType Screen

Figure 5.3-2 presents the Select User Type**,** Create Account**,** and Sign In screens. The user selection screen is shown upon first use and allows children to choose whether they are under or above six years of age. The interface is split into two cartoon-style sections with distinct visual cues: one for younger children and one for older users. Upon selection, animated effects highlight the chosen option. Older users proceed to either create an account or log in, using screens with large, clear input fields and friendly space-themed backgrounds. Validation feedback is simple and direct to prevent confusion, and loading screens include animated rockets for visual clarity during processing.

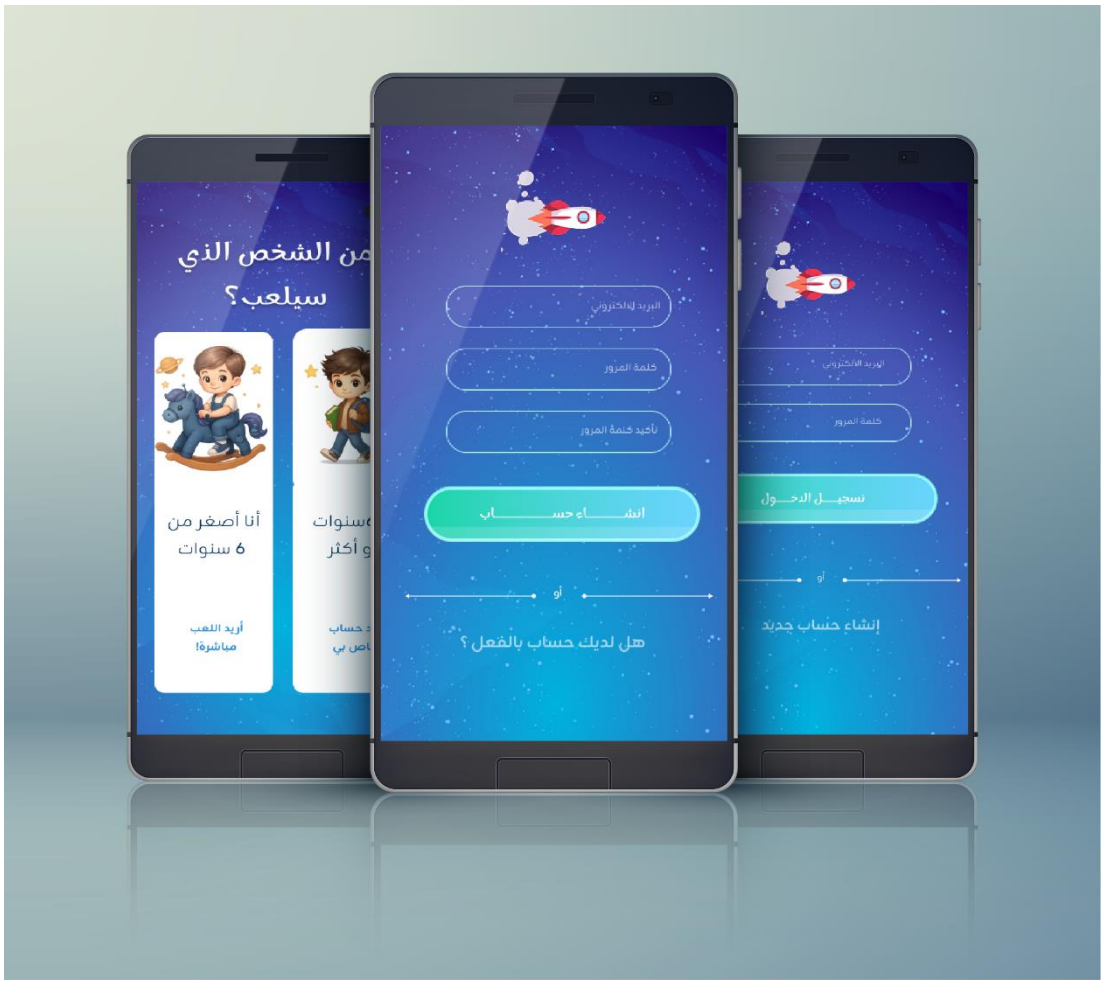


Figure 6.3-2 User Registration and Login Screens

- SelectCharacterScreen & WriteNameScreen

Figure 5.3-3 displays the Write Name and Select Character screens. These screens personalize the experience by allowing children to enter their names and choose an avatar. The interface uses playful space elements such as smiling moons and stars. Once the name is entered, the child is presented with a set of diverse cartoon characters to choose from. Each character is uniquely styled to reflect inclusivity and space-themed design. The final selection leads the child into a fully personalized journey across the app.

Figure 6.3-3 SelectCharacterScreen & WriteNameScreen

- Home Screen & Drawer

Figure 5.3-4 shows the Home Screen and Drawer Menu, which form the core navigation system of the application. The home screen is designed as a colorful space map, where each planet represents a different learning category or mini-game (e.g., letters, colors, animals). The visual layout encourages exploration and interaction, with animated space backgrounds, orbiting effects, and clearly labeled "Play" buttons on each planet. This layout simulates a playful adventure, allowing children to feel as if they are traveling through space to discover knowledge.

Planets are connected by glowing paths, guiding the child intuitively from one activity to another. When a planet is completed, visual rewards such as stars or medals are displayed to encourage progress and reinforce motivation. The design avoids clutter by spacing out the planets logically and using distinct colors for easy recognition.

The drawer menu complements the home screen by offering structured navigation. It can be accessed through a swipe or a menu button located in the top corner. Once opened, the drawer displays the child’s avatar, name, and email at the top, reinforcing the personal nature of the app. Below that, icons and text labels guide the user to key sections: Home, Profile, Share App, About Us, Rate Us, and Logout. The interface uses large, friendly buttons with vibrant colors and easily recognizable icons to suit the target age group. The drawer automatically closes when the user taps outside or navigates back, maintaining a seamless user experience.

Figure 6.3-4 Home Screen & Drawer

- Profile Screen & EditProfileScreen

Figure 5.3-5 presents the Profile Screen and Edit Profile Screen, which allow children to manage their account settings in an accessible and engaging way. The profile screen displays the user's selected character, name, and account information. Several large buttons beneath allow the child to perform key actions such as editing their profile, adding another child, muting/unmuting the application sound, or logging out.

The design features soft space-themed visuals—clouds, stars, and rockets—to maintain visual continuity with the rest of the app. When the user taps "Edit Profile," they are taken to a dedicated screen where they can update their avatar or name.

A small overlay icon on the avatar allows them to choose a new character, while an input field enables name changes. A clearly labeled "Save Changes" button at the bottom confirms their edits.

A confirmation dialog is shown if the child chooses to log out, helping to prevent accidental logouts. The interface also includes a feature to mute sounds with a single tap, ensuring flexibility for use in different environments. These screens are designed with simplicity and responsiveness in mind, allowing even young users to navigate and manage their profiles without adult assistance.

Figure 6.3-5 Profile Screen & EditProfileScreen

6.4 Games UI

- **Coloring Game Screen**

Figure 6.4-6 illustrates the Coloring Game, where children select from a palette of bright, appealing colors located at the bottom of the screen. Fill colors by clicking on the uncolored item in a simple yet attractive illustration, commonly animals or objects designed to stimulate creativity and fine motor skills. Progress is visually represented by stars at the top, serving as rewards for completing each coloring task and encouraging children to finish the whole picture. The interface balances playfulness with clarity: buttons for editing, muting, and logging out are prominently displayed for easy access. Background music and sound effects (if any) are subtle to maintain focus on coloring activities.

Figure 6.4-6 Coloring Game Screen

- Connection Game Screen

Figure 6.4-7 illustrates the Connection Game, which helps children develop cognitive linking and association skills. Players draw lines to connect related items, such as matching objects with their similar items. The interface features soft, calming background colors designed to minimize distractions, allowing children to focus on the task. Large, clearly labeled buttons ensure easy navigation and control, reducing confusion and enhancing usability. The game includes progressively harder levels, encouraging sustained engagement and gradual skill improvement. Feedback is immediate, with sound effects confirming correct matches, enhancing motivation and learning.

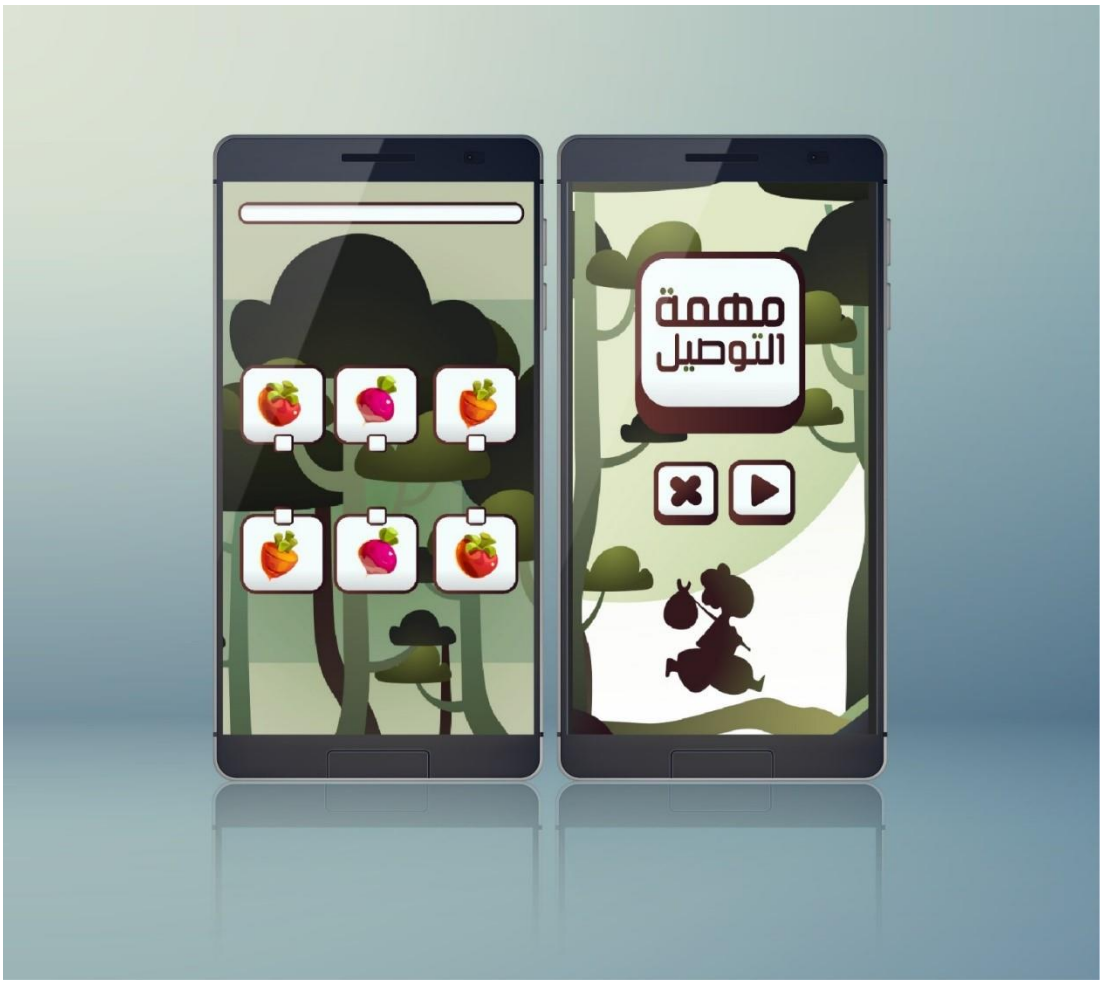


Figure 6.4-**7** Connection Game Screen

-Matching Cards Game Screen

Figure 6.4-8 illustrates the Matching Cards Game, aimed at improving memory and attention through a classic card-flipping mechanic. Children flip cards to find matching pairs of images, like fruits or shapes. The background uses soft tones to avoid visual overstimulation, and images on the cards are clear and colorful to aid recognition. The game progressively increases difficulty by adding more cards or similar-looking pairs to challenge cognitive skills. Large, user-friendly buttons provide smooth control for restarting levels or moving between stages. The game rewards players with visual effects and star ratings, reinforcing accomplishment and encouraging replay.

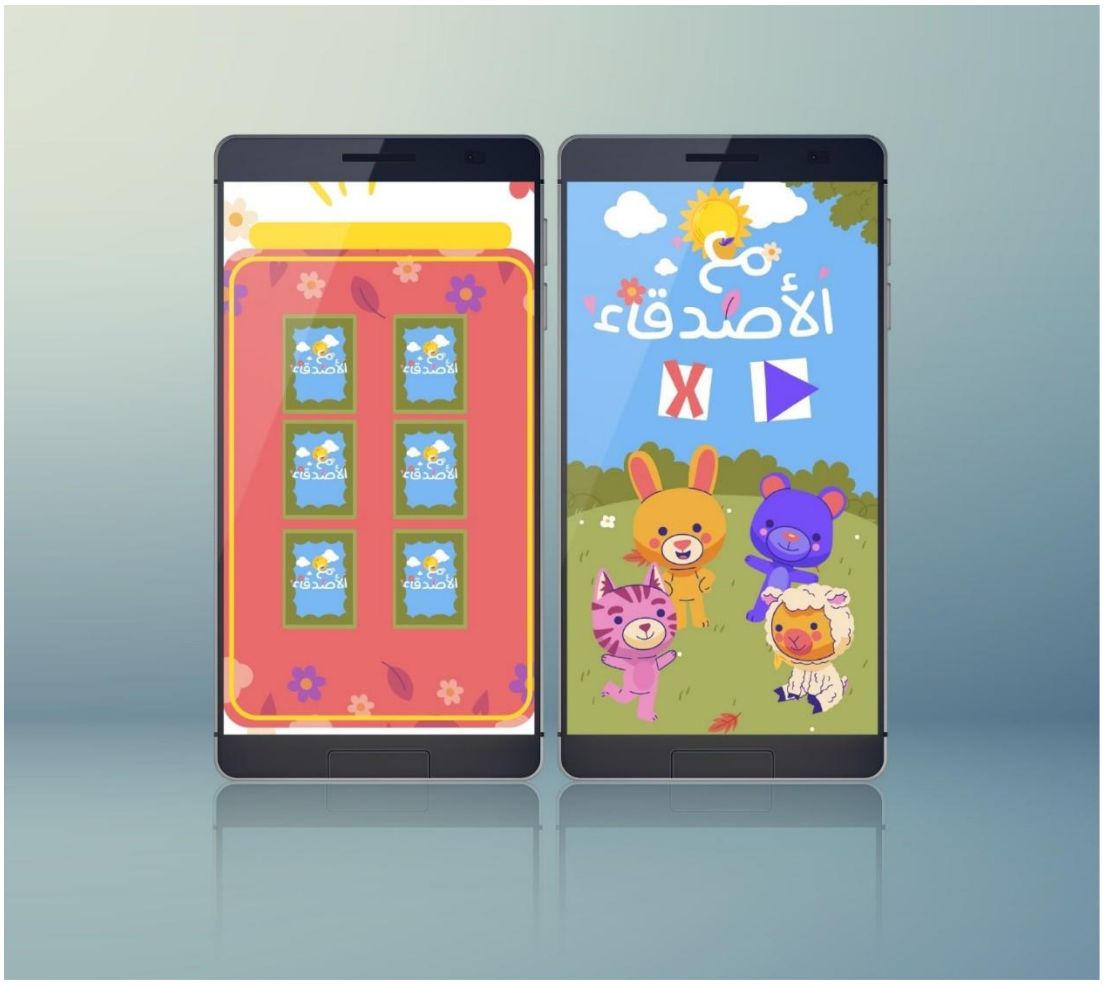


Figure 6.4- 8 Matching Cards Game Screen

- Rotation Puzzle Game Screen

Figure 6.4-9 illustrates the first Puzzle Game, designed to promote logical thinking, spatial awareness, and problem-solving. Children are presented with a divided image and must press each puzzle piece to its rotated position to reassemble the full picture. The interface uses soft colors and minimal distractions to help children focus, while a progress bar or star rating at the top provides real-time feedback on their advancement. This visual feedback motivates the child and instills a sense of achievement upon completion. Puzzle images are carefully chosen to be age-appropriate and engaging.

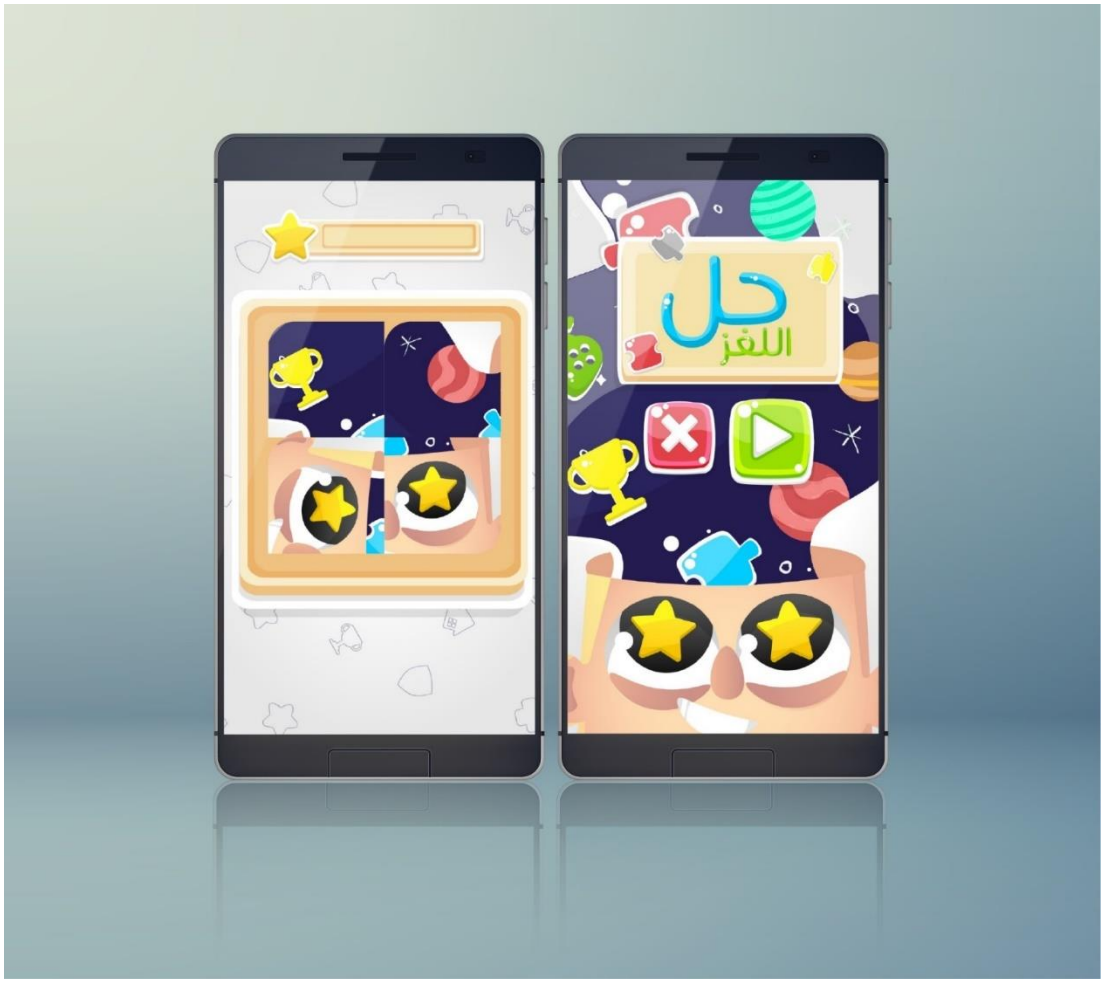


Figure 6.4 - 9 Puzzle Game Screen

- **Drag-to-Position Game Screen**

Figure 6.4-10 presents the Shadow Matching Game, which helps children improve their visual recognition skills. In this game, children are asked to match each image with its correct shadow by dragging the image to the place of the right shadow. The design is inspired by a space theme with engaging colors and backgrounds, making the experience more enjoyable and fitting the overall app theme. Each successful match rewards the player and encourages further progress.

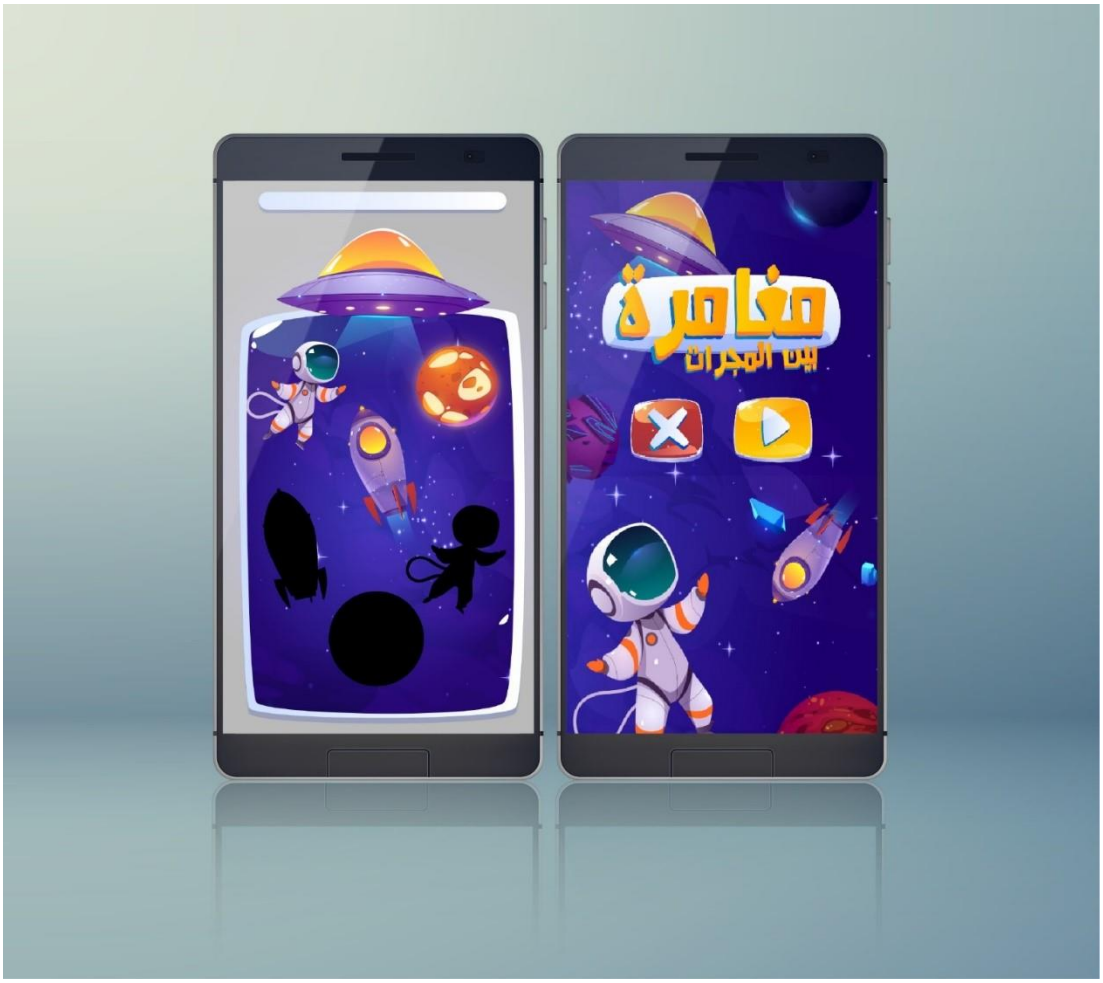


Figure 6.4-10 Drag-to-Position Game Screen

6.5 ER Diagram

The following Figure 6.5-1 shows the relationships between the tables:

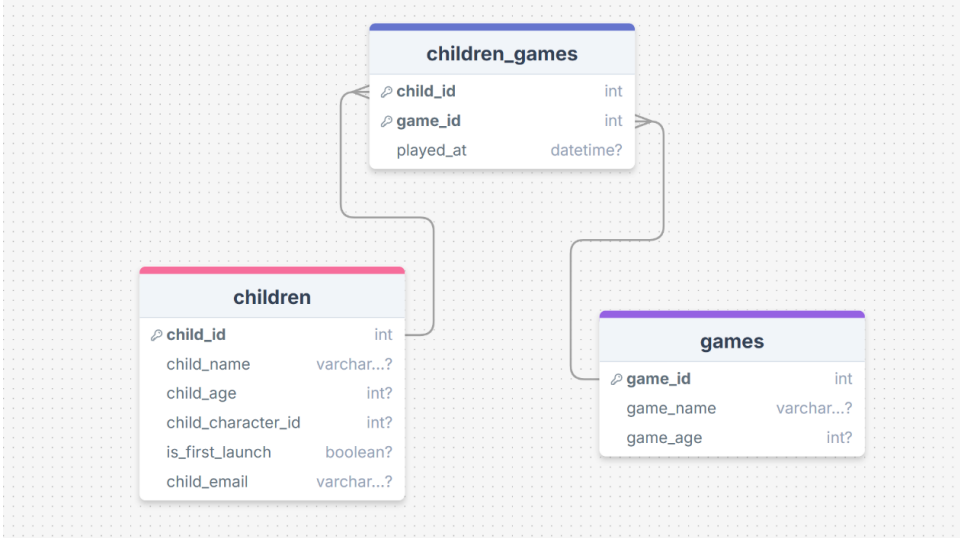


Figure 6.5-1 relationships between the tables

Table 6.5.1 Children Table

Field Name	Data Type	Description
child_id	INT(PK)	Unique identifier for the child (primary key).
child_name	VARCHAR (100)	The name of the child. It can be entered manually or assigned automatically depending on age.
Child_age	INT	The age of the child, is used to determine experience flow and required data.
child's_character_id	INT	The ID of the character selected by the child within the game.
is_first_launch	BOOLEAN	Indicates whether this is the child's first time using the app.
child_email	VARCHAR (100)	The child's email address, is only required if age ≥ 6. Can be null for guests or younger children.

Table 6.5.2 Games Table

Field Name	Data Type	Description
game_id	INT(PK)	Unique identifier for the game (primary key).
game_name	VARCHAR (100)	The name/title of the game as shown to the child.
game_age	INT	recommended age to play this game
game_id	INT (FK)	Last or current game associated with the child. References Game.game_id.

Chapter 7

Implementation

Chapter 7
Implementation

7.1 Introduction

This chapter presents the process of transforming designs and plans into a working system. The section covers development environment setup, database design, user interface construction, integration of educational games, and thorough testing to ensure smooth operation.

7.2 Implementation Flowchart

The following flowchart summarizes the main operational steps and user journey from first launch to accessing educational content:

1. **Splash and Welcome Screen**
    - Display introductory visuals and animations upon starting the app.
2. **Age Selection Screen**
    - Request the user to select age group: under 6 years or 6 years and above.
3. **User Pathways**
    - If under 6 years:
        - Proceed directly to character selection screen.
        - Move to the main home screen (planets and games).
    - If 6 years or above:
        - Prompt for account creation or login.
        - After successful registration/login, request name entry and character selection.
        - Navigate to the main home screen.
4. **Home Screen**
    - Display available educational games as interactive planets.
    - Enable navigation to profile, settings, and additional features through a side menu.
5. **Game Selection**

- Allow selection of a game (planet).
- Launch the selected educational game.

6. **Progress Tracking**
   - Save user progress and scores after each game.
   - Return to home screen or proceed to the next game.
7. **Additional Actions**
   - Provide options to view or edit profile, log out, or exit the app at any time.

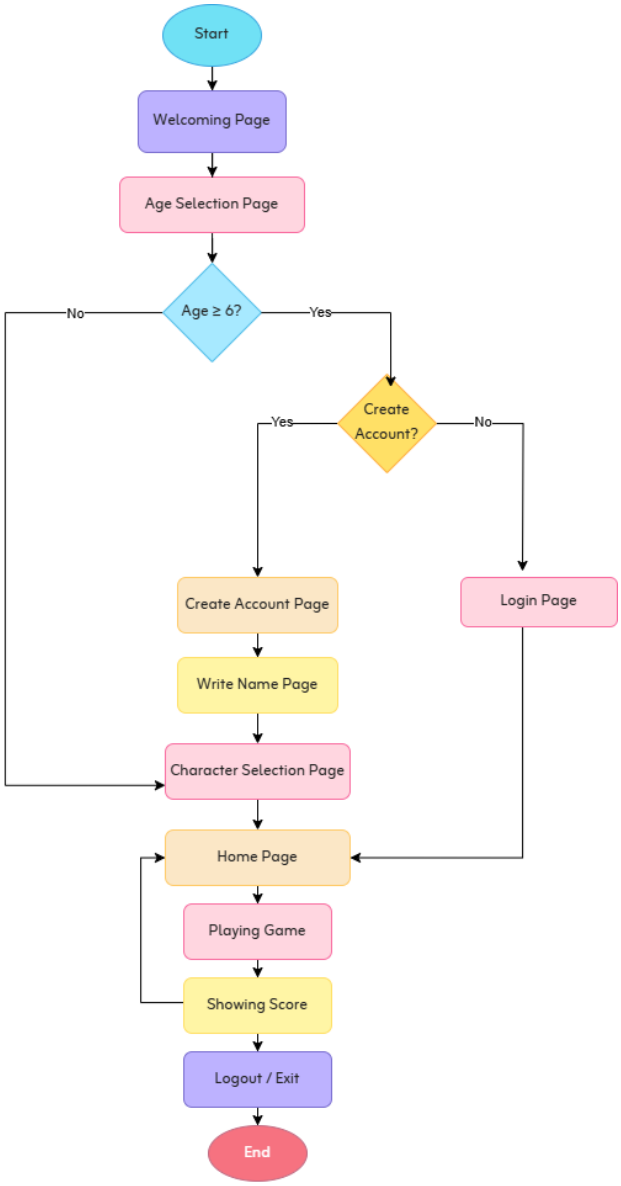


Figure 7.2-1 Flowchart

7.3 Implementation Codes

To provide a clearer understanding of the project, a portion of the project's code is showcased, divided into two sections: the Android application side and the Unity games side. Implementation Codes of Android Application

7.3.1 Implementation Codes of Android Application

Authentication Functions

- Create Account

- Display a loading overlay during registration.
- Register new user in Firebase Authentication using email and password.
- On success:
    - Save the user’s email locally (SharedPreferences).
    - Show a confirmation message (Toast).
    - Navigate to the screen for entering a name and selecting a character.
- On failure:
    - Show an error message based on the error type (weak password, duplicate email, invalid format, or connectivity issue).

- Reset the UI to allow another attempt.

```
private fun createAccount(email: String, password: String) {
      showLoading()
      auth.createUserWithEmailAndPassword(email, password)
          .addOnCompleteListener { task ->
              if (task.isSuccessful) {
                  saveEmailToPrefs(email)
                  Toast.makeText(this, "تم إنشاء الحساب بنجاح", Toast.LENGTH_SHORT).show()
                  startActivity(Intent(this, WriteNameActivity::class.java))
                  finish()
              } else {
                  handleCreateAccountErrors(task.exception)
                  hideLoading()
              }
          }
  }
```

Figure 7.3-1 Create Account Function

- Sign In

- Activate a loading overlay during sign-in.
- Check the provided email in Firestore to confirm existence.
- When the email exists:
    - Authenticate user with email and password via Firebase.
    - Retrieve user data (name, character ID, email, user type) from Firestore.
    - Save user data locally to personalize the experience.
    - Move to the correct home screen.
- On error:
    - Display a suitable error message (invalid credentials, unregistered email, or connection issue).
    - Reset the UI for another attempt.

```
private fun signInUser(email: String, password: String) {
    showLoading()

    val db = FirebaseFirestore.getInstance()
    db.collection("users").whereEqualTo("email", email).get()
        .addOnSuccessListener { result ->
            if (result.isEmpty) {
                emailFieldLayout.error = "البريد الإلكتروني غير مسجل"
                hideLoading()
            } else {
                FirebaseAuth.getInstance()
                    .signInWithEmailAndPassword(email, password)
                    .addOnCompleteListener { task ->
                        if (task.isSuccessful) fetchUserData()
                        else handleLoginErrors(task.exception)
                    }
            }
        }
        .addOnFailureListener {
            Toast.makeText(this, "فشل الاتصال بقاعدة البيانات", Toast.LENGTH_SHORT).show()
            hideLoading()
        }
}
```

Figure 7.3-2 Sign In Function

- Logout

- Show a confirmation dialog before logout.
- On confirmation:
    - Temporarily store the initial launch flag.
    - Clear all local user data.
    - Sign out from Firebase.
    - Navigate to the welcome screen, clearing all previous screens from memory.
- On cancel: close the dialog only.

```
private fun showLogoutDialog() {
    val view = layoutInflater.inflate(R.layout.dialog_logout, null)
    val dialog = AlertDialog.Builder(this).setView(view).create()

    view.findViewById<Button>(R.id.doneBtn).setOnClickListener {
        val prefs = getSharedPreferences("user_prefs", Context.MODE_PRIVATE)
        val isFirstLaunch = prefs.getBoolean("is_first_launch", false)
        prefs.edit().clear().putBoolean("is_first_launch", isFirstLaunch).apply()

        FirebaseAuth.getInstance().signOut()
        dialog.dismiss()
        startActivity(Intent(this, HelloActivity::class.java).apply {
            flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
        })
        finish()
    }

    view.findViewById<Button>(R.id.cancelBtn).setOnClickListener { dialog.dismiss() }

    dialog.window?.setBackgroundDrawableResource(android.R.color.transparent)
    dialog.show()
}
```

Figure 7.3-3 Logout Function

Data Storage Functions

- Save User

- Retrieve user's name, email, selected character, and default game ID from SP.
- Prepare user data and store it in Firestore using UID as the document ID.
- On success: mark onboarding as complete and navigate to the home screen.
- On failure: show an error message and allow retry.

```
private fun saveUserToFirestore() {
    val name = getUserPrefs("user_name", "")
    val email = getUserPrefs("user_email", "")
    val gameId = 0

    FirebaseAuth.getInstance().addAuthStateListener { auth ->
        auth.currentUser?.let { user ->
            val userData = hashMapOf(
                "uid" to user.uid,
                "email" to email,
                "name" to name,
                "character_id" to selectedCharacterId,
                "gameId" to gameId
            )

            FirebaseFirestore.getInstance().collection("users").document(user.uid)
                .set(userData)
                .addOnSuccessListener {
                    prefs.edit().putBoolean("user_completed_onboarding", true).apply()
                    navigateToHome()
                }
                .addOnFailureListener {
                    Toast.makeText(this, "Saving failed: ${it.message}", Toast.LENGTH_LONG).show()
                }
        } ?: Toast.makeText(this, "User not logged in", Toast.LENGTH_SHORT).show()
    }
}
```

Figure 7.3-4 Save User Function

- Update User Function

- Validate the entered name (cannot be empty).
- Save new name and character ID locally.
- For users under 6: store updates only on device and return to profile.
- For users 6 and above: update Firestore document with new name and character.
- On success: return to profile; on failure: show an error message.

```
private fun saveProfileChanges() {
    val name = if (nameEditText.visibility == View.VISIBLE)
        nameEditText.text.toString().trim()
    else
        nameTextView.text.toString().trim()

    if (name.isBlank()) {
        Toast.makeText(this, "يرجى إدخال الاسم", Toast.LENGTH_SHORT).show()
        return
    }

    val prefs = getSharedPreferences("user_prefs", Context.MODE_PRIVATE)
    prefs.edit().putString("user_name", name)
        .putInt("character_id", selectedImageId)
        .apply()

    if (prefs.getString("user_type", "under6") == "under6") {
        navigateToProfile()
    } else {
        val uid = FirebaseAuth.getInstance().currentUser?.uid ?: return
        val userData = mapOf("name" to name, "character_id" to selectedImageId)

        FirebaseFirestore.getInstance().collection("users").document(uid)
            .set(userData, SetOptions.merge())
            .addOnSuccessListener { navigateToProfile() }
            .addOnFailureListener {
                Toast.makeText(this, "تعذر الحفظ! حاول لاحقًا.", Toast.LENGTH_LONG).show()
            }
    }
}
```

Figure 7.3-5 Update User Function

Open Game Functions

- Start Game

- Update current game ID locally or in Firestore based on age group.
- Prepare data to send to Unity to select the correct game.
- Start the Unity game activity and receive game results after completion

```
private fun startGame(gameId: Int) {
        updateGameIdToCloud(gameId)

        val intent = Intent(requireContext(),
UnityPlayerGameActivity::class.java) "${getCardPrefix()} card number is: $gameId")
        startActivityForResult(intent, UNITY_REQUEST_CODE)
    }
```

Figure 7.3-6 Start Game Fun5ction

- update Game ID

- Read user type from local storage.
- For users under 6: save game ID locally.
- For users 6 and above: update game ID in Firestore using UID.
- Ensure user progress is synced across devices.

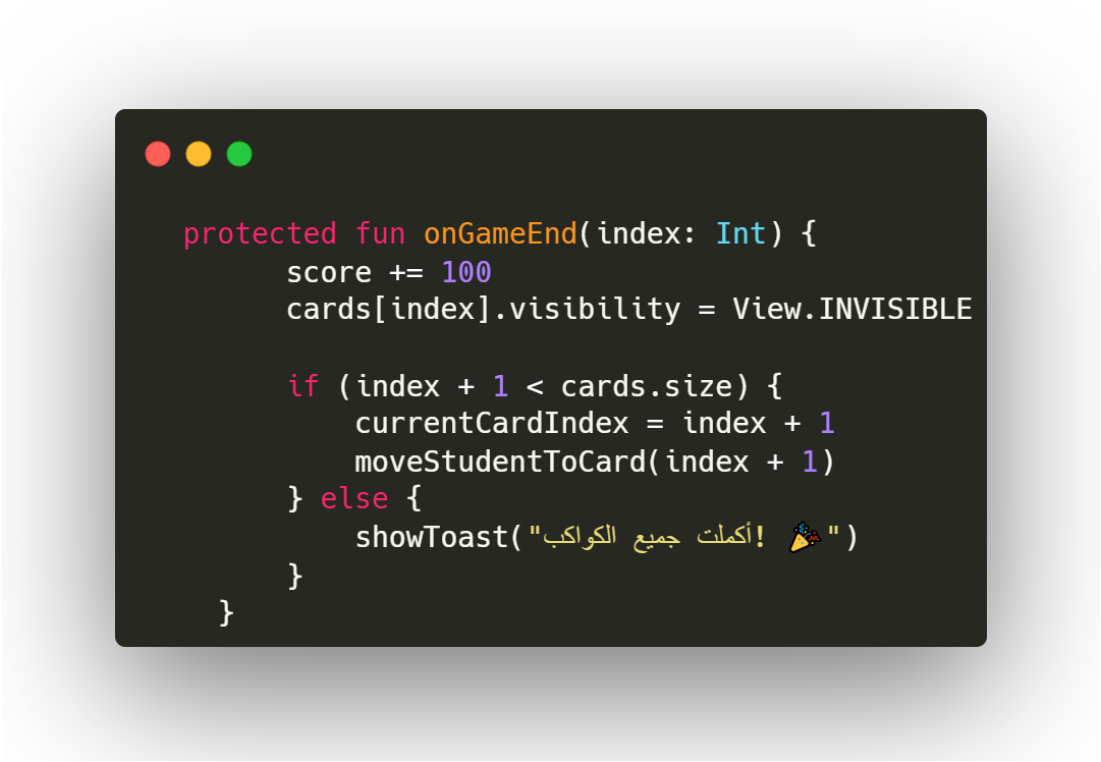
```
private fun updateGameIdToCloud(gameId: Int) {
        val prefs = requireActivity().getSharedPreferences("user_prefs",
Context.MODE_PRIVATE)= prefs.getString("user_type", "under6") ?: "under6"

        if (userType == "under6") {
            prefs.edit().putInt("game_id", gameId).apply()
        } else {
            FirebaseAuth.getInstance().currentUser?.uid?.let { uid ->
                FirebaseFirestore.getInstance().collection("users")
                    .document(uid).update("gameId", gameId)
            }
        }
    }
```

Figure 7.3-7 update GameId Function

- move Game

- Add points to the internal score after a game is completed.
- Hide the completed planet or card from the interface.
- If more planets are available, update the index and animate the character's movement to the next one.
- When all planets are finished, display a congratulatory message.



```
protected fun onGameEnd(index: Int) {
        score += 100
        cards[index].visibility = View.INVISIBLE

        if (index + 1 < cards.size) {
            currentCardIndex = index + 1
            moveStudentToCard(index + 1)
        } else {
            showToast("أكملت جميع الكواكب! 🎉")
        }
    }
```

.Figure 7.3-8 move Game Function

7.3.1 Implementation Codes of Unity Games

The development of the Unity-based games in the application combined engaging visual design with robust programming logic using C#. Each game was implemented through a set of scripts and user interface elements that provided an interactive and educational experience. Below, we detail the implementation and workflow for each game, with a corresponding figure to illustrate the main interface.

- Coloring Game

- On tap or click, apply the selected color to the chosen part of the drawing.
- Track the number of colored parts.
- When all parts are colored:
    - Trigger star animations as a reward.
    - Enable a button to return to the home screen.
- Store and manage the current color, colored parts, and total parts as variables.

highlights the color palette, progress stars, and an exit button for a smooth and friendly user experience.

```
void Update()
    {
        curColor = colorList[colorCount];

        var ray = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        RaycastHit2D hit = Physics2D.Raycast(ray, -Vector2.up);


        if (Input.GetButtonDown("Fire1"))
        {
            if (hit.collider != null)
            {
                SpriteRenderer sp = hit.collider.gameObject.GetComponent<SpriteRenderer>();

                if (sp != null)
                {
                    Debug.Log("Hit: " + hit.collider.name);
                    sp.color = curColor;

                    if (!coloredObjects.Contains(hit.collider.gameObject))
                    {
                        coloredObjects.Add(hit.collider.gameObject);
                        coloredParts++;
                        CheckCompletion();
                    }
                }
            }
        }
    }
```

Figure 7.3-9 Coloring Game

- Connection Game

- On card selection, enable dragging and show a visual connection line.
- When the card is dropped on a matching card, update the progress bar and disable matched cards.
- On mismatch, return the card to its original position and reset the connection line.

```
private void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        RaycastHit2D hit = Physics2D.Raycast(Camera.main.ScreenToWorldPoint(Input.mousePosition),
                Vector2.zero);

        if (hit.collider != null && hit.collider.gameObject == gameObject)
        {
            isDragging = true;

            Vector3 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
            mousePosition.z = 0f;
            lineRender.SetPosition(0, mousePosition);

            if (dragStartSound != null)
            {
                dragStartSound.Play();
            }
        }
    }
    }
}
```

Figure 7.3-10 Connection Game

- Memory Cards Game

- Generate pairs of cards, shuffle them, and arrange into a grid.

- On clicking two cards:

  - If matched, keep both face up.
  - If not matched, flip both back after a short delay.

- Limit to two cards being open at once.

Add animations and effects to encourage playing

```
void CreateCards()
    {
        for (int i = 0; i < spritePairs.Count; i++)
        {
            Card card = Instantiate(cardPrefab, gridTransform);
            card.SetIconSprite(spritePairs[i]);
            card.controller = this;
        }
    }


public void SetSelected (Card card)
    {
        if (card.isSelected == false)
        {
            card.Show();
            if (firstSelectedCard == null)
            {
                firstSelectedCard = card;
                return;
            }
            if (secondSelectedCard == null)
            {
                secondSelectedCard = card;
                StartCoroutine(CheckMatching(firstSelectedCard, secondSelectedCard));
                firstSelectedCard = null;
                secondSelectedCard = null;
            }
        }

    }
```

Figure 7.3-11 Memory Cards Game

- Drag-to-Position Puzzle

- On touch, track the selected puzzle piece.
- Move the piece with the finger, maintaining an offset for natural movement.
- On release near the correct spot, snap the piece into place.
- If not close, return the piece to its original position.
- Check for completion after each move

```
void Update()
    {
        if (Input.touchCount > 0 && !locked)
        {
            Touch touch = Input.GetTouch(0);
            Vector2 touchPosition = Camera.main.ScreenToWorldPoint(touch.position);

            switch (touch.phase)
            {
                case TouchPhase.Began:
                    if (GetComponent<Collider2D>() == Physics2D.OverlapPoint(touchPosition))
                    {
                        deltaX = touchPosition.x - transform.position.x;
                        deltaY = touchPosition.y - transform.position.y;
                    }
                    break;

                case TouchPhase.Moved:
                    if (GetComponent<Collider2D>() == Physics2D.OverlapPoint(touchPosition))
                    {
                        transform.position = new Vector2(touchPosition.x - deltaX, touchPosition.y - deltaY);
                    }
                    break;

                case TouchPhase.Ended:
                    if (Mathf.Abs(transform.position.x - iteamSilhouette.position.x) < 0.5f &&
                        Mathf.Abs(transform.position.y - iteamSilhouette.position.y) < 0.5f)
                    {
                        transform.position = new Vector2(iteamSilhouette.position.x, iteamSilhouette.position.y);
                        locked = true;
                    }
                    else
                    {
                        transform.position = new Vector2(initalPosition.x, initalPosition.y);
                    }

                    break;
            }
        }
    }
```

Figure 7.3-12 -Puzzle Game

- Rotation Puzzle

- In On tap, rotate puzzle pieces by 90° increments.
- Randomize initial orientation for each piece.

When all pieces are at 0°, trigger a win state and show victory animation.

```
    private void OnMouseDown()
    {
        if (!GameControl.youWin)
        {
            transform.Rotate(0f, 0f, 90f);
        }

    }
```

Figure 7.3-13 - Rotation Puzzle (Orientation Matching Logic)

7.3.2 Firebase Integration

To manage user data, authentication, and progress tracking, the *Majarat Al-Ma'rifa* application integrates with Firebase – a comprehensive backend-as-a-service platform. Firebase provides essential services such as Authentication and Cloud Firestore, both of which were critical in enabling real-time synchronization, secure sign-in, and personalized learning experiences.

- Firebase Authentication

- Use Firebase Authentication for sign-up and login (ages 6+).
- Store email, password, and unique user ID (UID) for each account.
- Manage user sessions and automatic logout as needed.

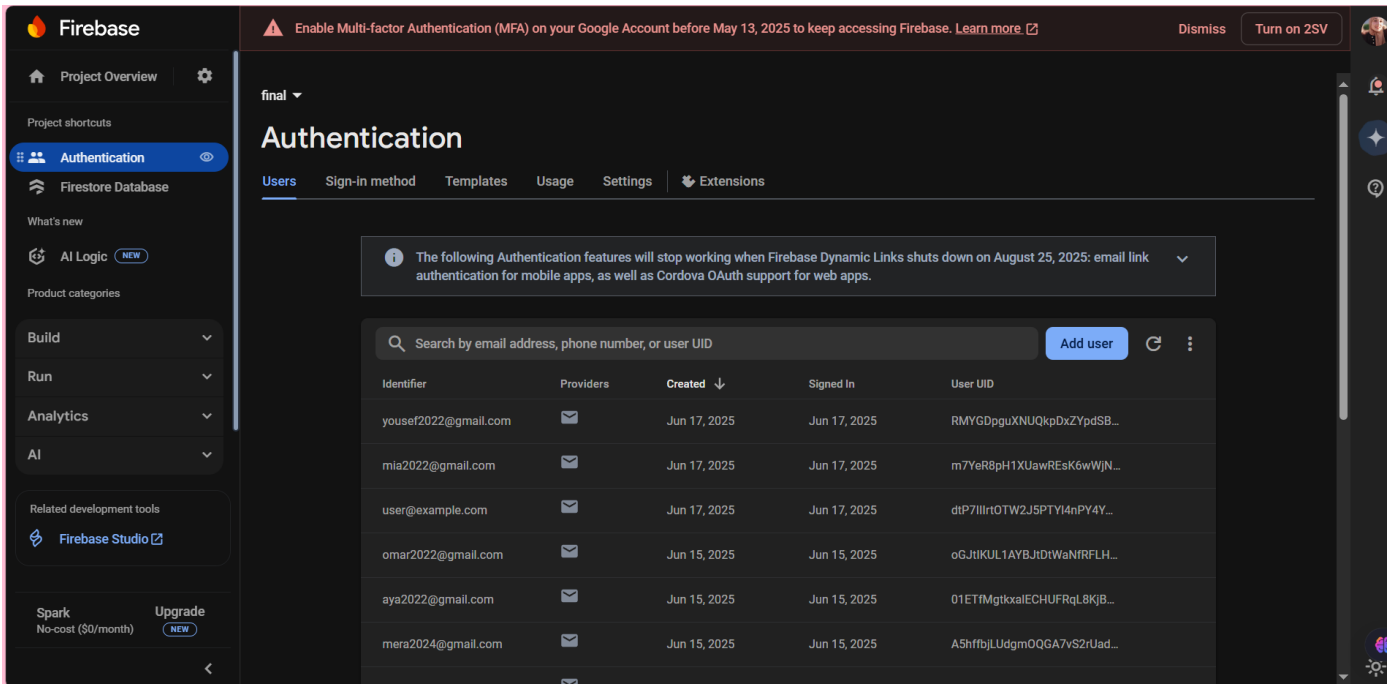


Figure 7.3-14 Firebase Authentication

- Cloud Firestore Database

- Store user data in the users collection (Firestore).
- Each document contains name, email, character ID, game ID, and UID.
- Enable real-time synchronization and instant data updates
- Update database automatically when user changes profile or progress.

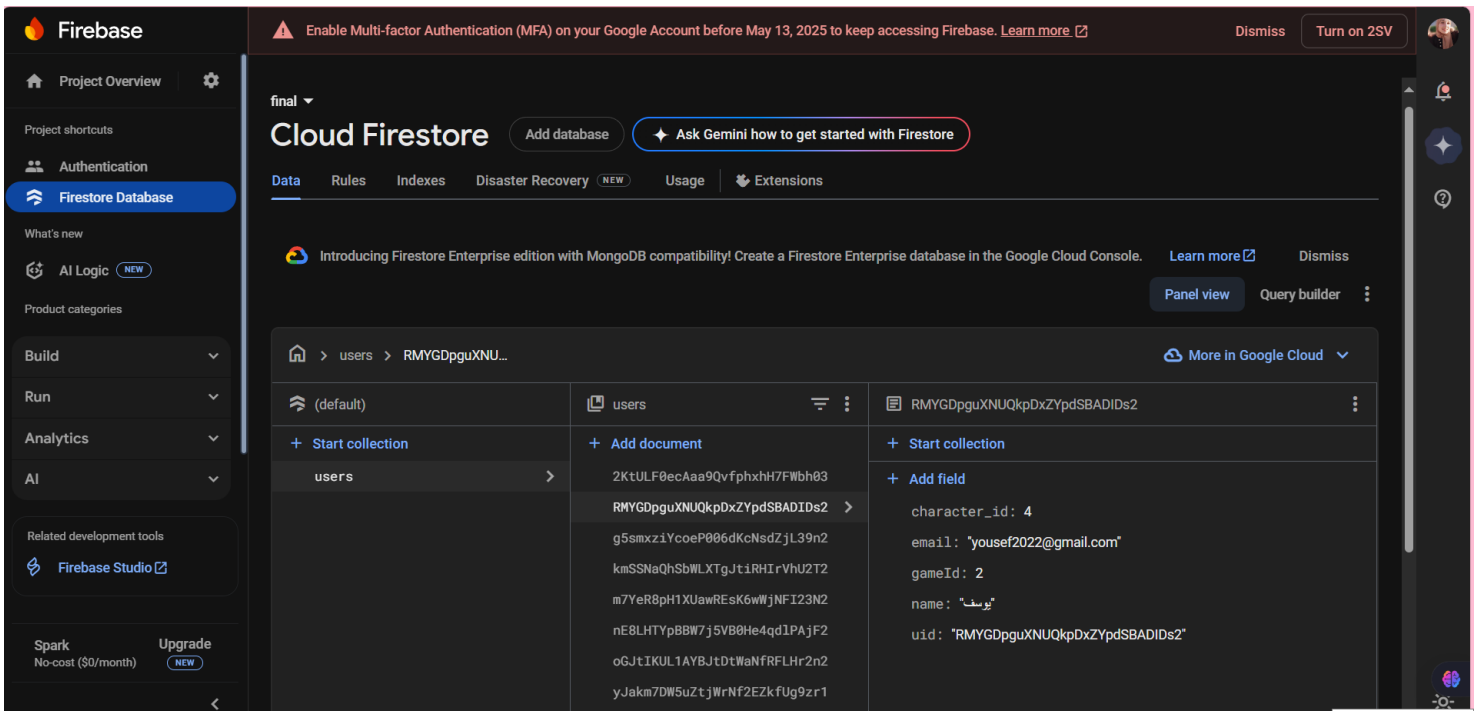


Figure 7.3-16 Cloud Firestore Database

- Firebase Dashboard and Usage Metrics

- Monitor Firestore operations and usage metrics via Firebase Dashboard.
- Track read and write activity for performance and optimization.
- Analyze user activity trends to guide improvements.

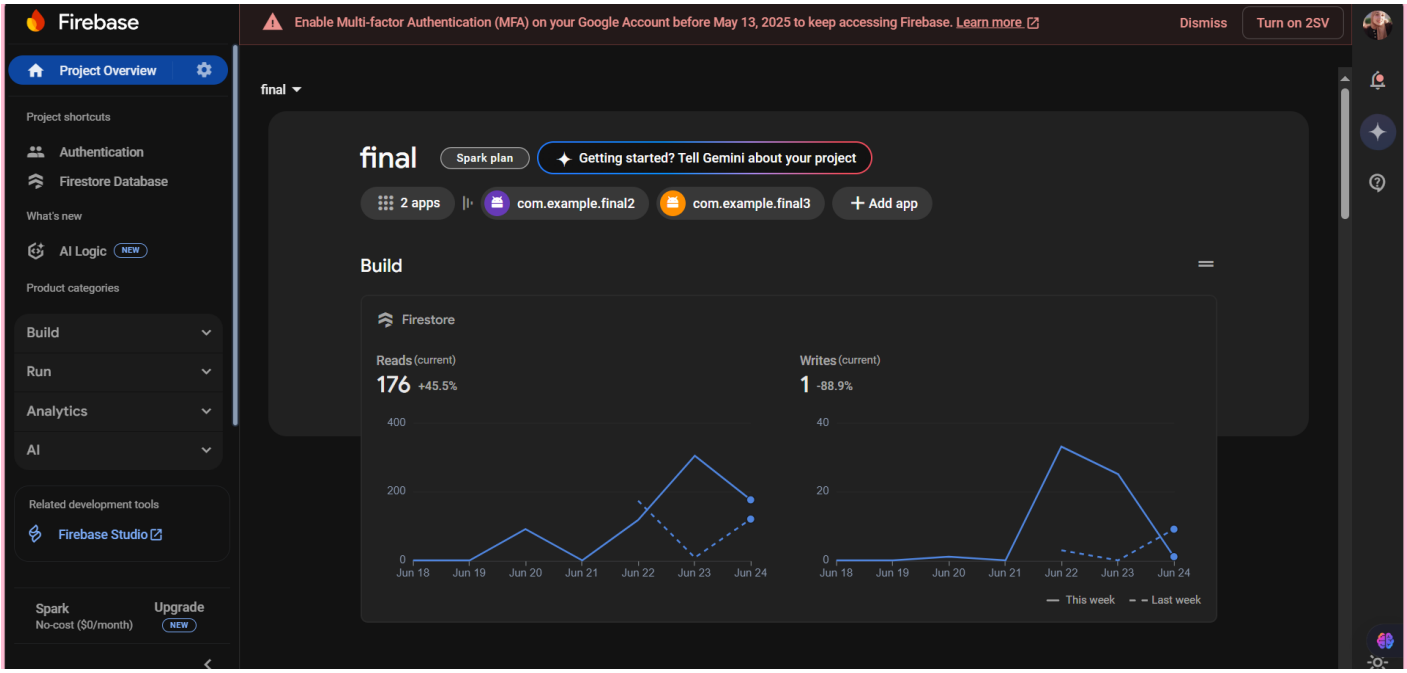


Figure 7.3-1 Firebase Dashboard and Usage Metrics

7.4 Sequence Diagrams for User Flows

To clarify the implementation logic for different age groups, sequence diagrams are provided. These diagrams illustrate the main interactions between the user, the application, and external modules for both simple and full flows.

SimpleFlow(Age<6):

The diagram below shows the process for users under 6 years. No account creation or cloud storage is required. Progress is saved locally after each game session.

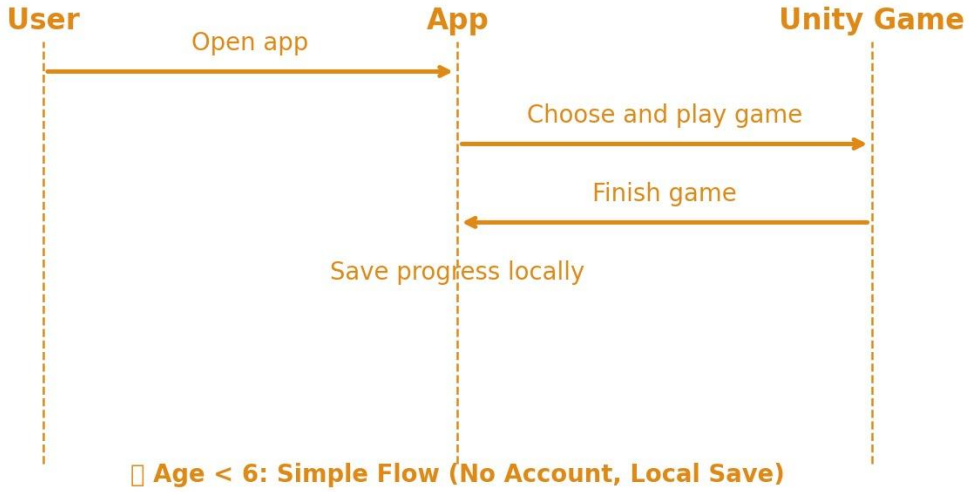


Figure 7.4-1 Simple Flow (Age < 6)

This flow consists of opening the application, selecting and playing a game, and saving the results locally on the device.

FullFlow(Age≥6):

The second diagram represents the process for users aged 6 years and above. This flow includes account creation, authentication, cloud data synchronization, and remote score tracking.

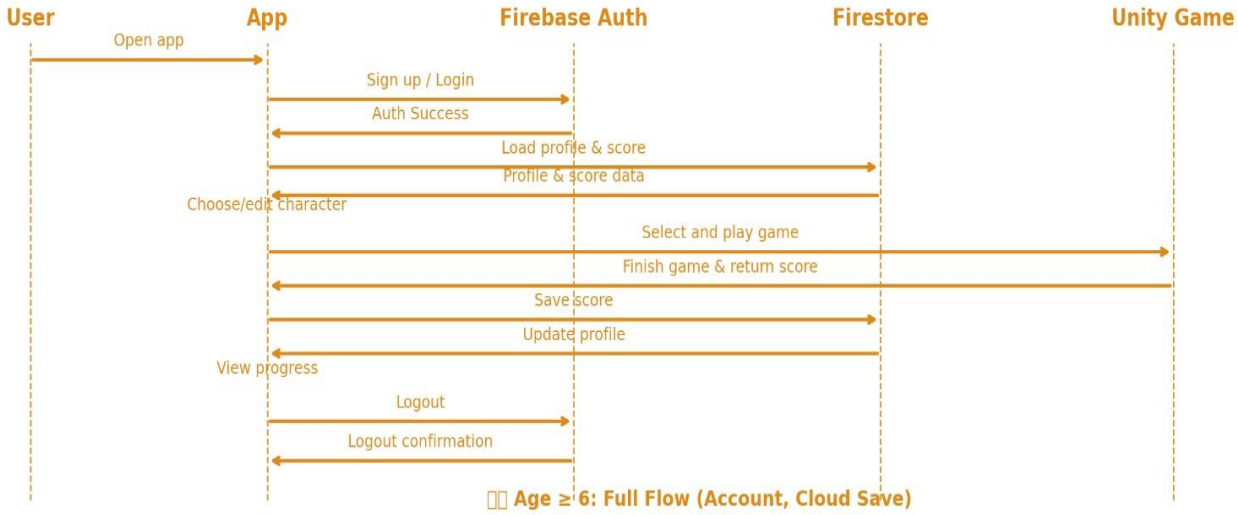


Figure 7.4-2 Full Flow (Age ≥ 6)

After signing up or logging in, profile data and scores are loaded from the cloud. Game results and any profile updates are saved back to the cloud. Viewing progress and logout are also managed through the cloud services to enable multi-device access.

Chapter 8

Testing

Chapter 8
Testing

8.1 Introduction

Testing is an essential phase in the development of any mobile application. It ensures that the app functions as intended and that users do not encounter any issues that could compromise their experience. For our educational app, we implemented various types of testing to confirm that the app is not only user-friendly and engaging but also safe for children. Through testing, we validated that every feature of the app, from the UI components to the navigation, operates smoothly as expected. In this chapter, we will explain the key types of testing we employed, including unit testing, integration testing, system testing, and user interface (UI) testing..

8.2 Unit Testing

We tested each screen individually, focusing on the logic and validations behind every function. For example, in the Sign-Up screen, we used the validateFields() function to test whether the user input meets the required conditions before account creation is allowed.

This function checks three main fields: email, password, and confirm password. We performed the following tests:

- If the email is empty, the function shows an error: “Email is required.”
- If the email format is invalid, it shows: “Invalid email format.”
- If the password is empty or less than 6 characters, it shows appropriate error messages.
- If the confirmation password is empty or doesn’t match the original password, error messages are also triggered.

We tested valid and invalid inputs to ensure that these error messages appear at the right times and prevent the user from continuing unless all fields are valid.

After entering the correct data in all fields, we confirmed that the form passed validation and moved forward to the next screen. The result of this unit test for the sign-up form is shown in Table 8.2.1.

Table 8.2.1 Unit Test – Sign-Up Validation

Field	**Input**	**Expected Result**	**Actual Result**
Email	(empty)	Show “Email is required”	Passed
Email	user@invalid	Show “Invalid email format”	Passed
Password	(empty)	Show “Password is required”	Passed
Password	123	Show “Password too short”	Passed
Confirm Password	(empty)	Show “Confirm password is required”	Passed
Confirm Password	123456 vs. 123123	Show “Passwords do not match”	Passed
All fields valid	user@example.com / 123456 / 123456	Proceed to the next screen	Passed

8.3 Integration Testing

Integration testing is the process of verifying how different components of the application interact when they are connected. In our case, we used a bottom-up approach, where we combined and tested small functional parts step by step until the entire system was integrated.

For example, we tested the connection between the sign-up screen, the Firebase authentication module, and the Firestore database. After the user signs up, their data (name, email, and selected character) should be saved both locally and in the cloud. We verified that once the sign-up is successful, the app transitions to the home screen and displays the correct user information.

Another integration point we tested was between the character selection screen and the profile update logic. After selecting a character, the app should reflect this choice on the user profile and save it permanently.

All integration points between screens, database updates, and user navigation were tested, and the results were successful with no critical issues found.

8.4 System Testing

System testing is a full end-to-end test of the complete app to ensure that everything works together as one unit. We tested the whole educational game application as if we were real users, starting from account creation and continuing through character selection, gameplay, profile updates, and logout.

We created different user types (under 6 and above 6) and verified that the system responds differently for each type, as required. For example:

- Children under 6 were allowed to play without authentication and only store their data locally.
- Children above 6 had to sign in and their progress was stored in Firestore.

We verified that music plays in the background during navigation, and that all transitions between screens work smoothly without crashing. Every function was tested against the initial requirements, and the actual results matched the expected behavior.

Table 8.4.1 System Test – Complete User Journey

Scenario	Steps Performed	Expected Result	Actual Result
Full registration flow	Sign up → Select character → Save to Firestore → Navigate	Profile appears with correct info	Passed
Game flow	Tap on planet → Open Unity game → Return with score	Progress updates and next level unlocked	Passed
Profile update	Change name & character → Save	Profile data updated in Firestore	Passed

8.5 UI Testing

For our app Mujaraat Al-Ma'arifah (The Knowledge Galaxy), which focuses on educational games for children, UI testing is a crucial step to ensure a seamless and engaging user experience. UI testing in the context of our app involves validating the behavior and correctness of the user interface components specific to educational functionalities for children.

Using frameworks like Espresso, we've been able to automate UI tests to simulate user interactions and verify that Mujaraat Al-Ma'arifah UI elements work as intended. These tests can be run on Android devices or emulators to evaluate app performance and responsiveness across different screen sizes and resolutions.

Testing Scenario:

```kotlin
@Test
fun testValidateFields_EmptyEmail() {
    val scenario = ActivityScenario.launch(CreateAccountActivity::class.java)
    onView(withId(R.id.create_emailInput)).check(matches(isDisplayed()))
    onView(withId(R.id.create_emailInput)).perform(typeText( stringToBeTyped: ""), closeSoftKeyboard())
    onView(withId(R.id.create_passwordInput)).perform(typeText( stringToBeTyped: "password123"), closeSoftKeyboard())
    onView(withId(R.id.create_confirmPasswordInput)).perform(typeText( stringToBeTyped: "password123"), closeSoftKeyboard())

    onView(withId(R.id.createAccountBtn)).perform(click())

    onView(withId(R.id.create_emailFieldLayout))
        .check(matches(hasDescendant(withText( text: "البريد الالكتروني مطلوب"))))
}

@Test
fun testValidateFields_InvalidEmail() {
    val scenario = ActivityScenario.launch(CreateAccountActivity::class.java)
    onView(withId(R.id.create_emailInput)).check(matches(isDisplayed()))

    onView(withId(R.id.create_emailInput)).perform(typeText( stringToBeTyped: "invalid-email"), closeSoftKeyboard())
    onView(withId(R.id.create_passwordInput)).perform(typeText( stringToBeTyped: "password123"), closeSoftKeyboard())
    onView(withId(R.id.create_confirmPasswordInput)).perform(typeText( stringToBeTyped: "password123"), closeSoftKeyboard())

    onView(withId(R.id.createAccountBtn)).perform(click())

    onView(withId(R.id.create_emailFieldLayout))
        .check(matches(hasDescendant(withText( text: "صيغة البريد الالكتروني غير صحيحة"))))
}
```

Figure 8.5-1 Testing Scenario 1

```kotlin
@Test
fun testValidateFields_ShortPassword() {
    val scenario = ActivityScenario.launch(CreateAccountActivity::class.java)
    onView(withId(R.id.create_passwordInput)).check(matches(isDisplayed()))

    onView(withId(R.id.create_emailInput)).perform(typeText( stringToBeTyped: "user@example.com"), closeSoftKeyboard())
    onView(withId(R.id.create_passwordInput)).perform(typeText( stringToBeTyped: "pass"), closeSoftKeyboard()) // كلمة مرور قصيرة جداً
    onView(withId(R.id.create_confirmPasswordInput)).perform(typeText( stringToBeTyped: "pass"), closeSoftKeyboard())

    onView(withId(R.id.createAccountBtn)).perform(click())

    onView(withId(R.id.create_passwordFieldLayout))
        .check(matches(hasDescendant(withText( text: "كلمة المرور قصيرة جدًا"))))
}

@Test
fun testValidateFields_PasswordMismatch() {
    val scenario = ActivityScenario.launch(CreateAccountActivity::class.java)
    onView(withId(R.id.create_confirmPasswordInput)).check(matches(isDisplayed()))

    onView(withId(R.id.create_emailInput)).perform(typeText( stringToBeTyped: "user@example.com"), closeSoftKeyboard())
    onView(withId(R.id.create_passwordInput)).perform(typeText( stringToBeTyped: "password123"), closeSoftKeyboard())
    onView(withId(R.id.create_confirmPasswordInput)).perform(typeText( stringToBeTyped: "differentpassword"), closeSoftKeyboard()) // كلمات

    onView(withId(R.id.createAccountBtn)).perform(click())

    onView(withId(R.id.create_confirmPasswordFieldLayout))
        .check(matches(hasDescendant(withText( text: "كلمات المرور غير متطابقة"))))
}
```

Figure 8.5-2 Testing Scenario 2

In Figures 8.5.1 and 8.5.2, the tests focus on email, password, and confirm password fields to ensure that invalid data prevents users from proceeding. The tests include:

- Email Field Validation:

  If the email field is empty or contains an invalid format, the app should show "البريد الإلكتروني مطلوب" ("Email is required") or "صيغة البريد الإلكتروني غير صحيحة" ("Invalid email format").

- Password Field Validation:

  If the password is too short (less than 6 characters), the app should display "كلمة المرور قصيرة جدًا" ("Password too short").

- Confirm Password Validation:

  If the confirmation password does not match the original password, the app should show "كلمات المرور غير متطابقة" ("Passwords do not match").

These tests ensure that the user cannot proceed unless all fields contain valid data.

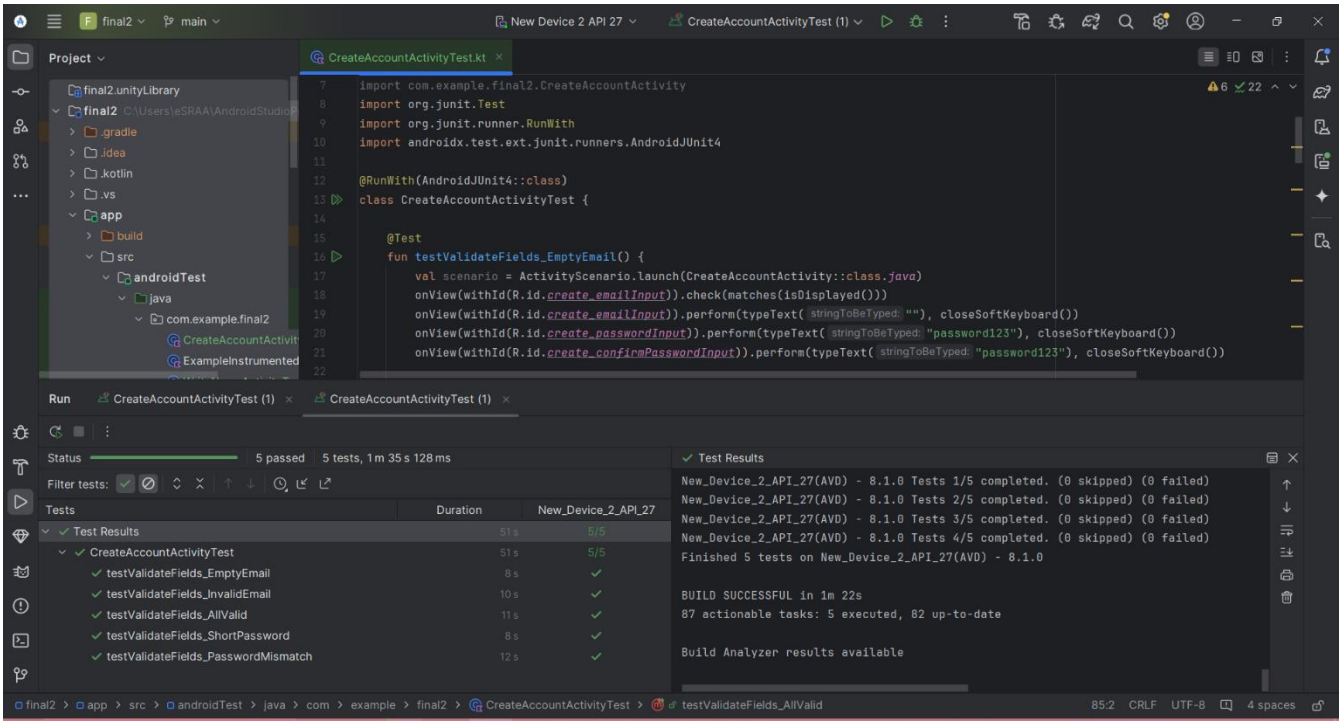


Figure 8.5-3 UI Test Results

Figure 8.5.3 shows the result of UI testing for validating the account creation screenThe test simulates the user entering valid data in the email, password, and confirm password fields

- Valid email: "user@example.com"
- Valid password: "password123"
- Matching confirmation password: "password123"

Upon entering the correct information, the app allowed the user to proceed to the next screen, successfully passing the test. This shows that the form properly validated the user inputs and moved the user to the next screen when all fields were valid.

This test ensures that the form passes validation and moves the user to the next screen when all fields contain valid data.

Chapter 9

Future Works

Chapter 9
Future Works

As *Majarat Al-Ma'rifa* continues to grow, several future enhancements are envisioned to enrich its educational value, improve user experience, and expand technological capabilities. Building on the current foundation, the app can be developed further by focusing on accessibility, engagement, and adaptability.

One key area of development is the integration of animated storytelling and audio, allowing content to be delivered through engaging characters and narration, especially for early learners. The introduction of a family mode with multi-user support and advanced analytics will personalize learning and assist parents and educators in tracking progress.

Technologies like Augmented Reality (AR) and AI-based learning assistants promise to make learning more interactive and intelligent. To reach a wider audience, multilingual support and accessibility features such as subtitles and text-to-speech will be critical.

Other planned improvements include online content updates for fresh material, offline access for flexibility, and a motivational scoring system to encourage consistent engagement. Collaborative learning modes will promote teamwork, and expanding the app's content to cover more subjects like science and health will support well-rounded learning.

Finally, adding parental and teacher tools and establishing a safe community space will foster support and safe interaction, making the app not only educational but also socially enriching.

These future plans aim to transform *Majarat Al-Ma'rifa* into a comprehensive, inclusive, and forward-thinking learning platform for children.

Chapter 10

Conclusion

Chapter 10
Conclusion

The *Majarat Al-Ma'rifa* project aimed to create a fun and educational mobile application for children, combining learning with a space-themed adventure. The app allows children to explore different topics through interactive games and engaging visuals designed to suit their age and interests.

Throughout the project, focus was placed on designing a simple, colorful, and child-friendly interface. Features such as personalized characters, age-based navigation, and easy controls were used to enhance the learning experience.

The application successfully demonstrates how mobile technology can support early childhood education. While the current version provides a solid foundation, future improvements—such as storytelling, voice narration, and performance tracking—can make the app more impactful and inclusive.

In summary, *Majarat Al-Ma'rifa* offers an enjoyable way for children to learn independently and represents a promising step toward modern, gamified education.

References

[1] M. A. B.-A .& .K. Hornbæk ،Old wine in new bottles or novel challenges: A critical analysis of empirical studies of user experience ،Vancouver, Canada: ACM (Association for Computing Machinery) .2011 ،

[2] V. Rideout ،The Common Sense Census: Media Use by Kids Age Zero to Eight ،San Francisco, California :Common Sense Media .2017 ،

[3] R. M. Z. R. M. G. K. H. G. M. K. R. J. M. K. L. Hirsh-Pasek ،Putting education in ‘educational’ apps: Lessons from the science of learning ، Psychological Science in the Public Interest .2015 ،

[4] U. G. F. D. F. R. Victoria J. Rideout ،Generation M²: Media in the Lives of 8- to 18-Year-Olds ،Menlo Park, California: Henry J. Kaiser Family Foundation ، .2010

[5] H. T. S. W. Anjar Sari ،Mobile Game Prototype for Fractional Concept in Elementary School ،KnE Social Sciences .2024 ،

[6] N. F. A. N. I. O. Z. Z. Nur Fadzilah Ahmad ،The effect of mobile application to promote learning English for primary school students ،Advances in Mobile Learning Educational Research .2024 ،

[7] K. L. R. F. Natalia Kotserkova ،Selecting educational apps for preschool children: How useful are website app rating systems? ،British Journal of Educational Technology .2022 ،

[8] N. R. K. R. S. R. Sharmilan Sureshwaran ،Digital Learning For Kids ، International Journal of Advanced Research and Publications .2022 ،

[9] F. Aydoğdu ،Augmented reality for preschool children: An experience with educational contents ،British Journal of Educational Technology .2022 ،

[10] P. C.-L. M. A. G. L. G.-P. J. M. A.-P. R. T. A. A. n ،Between level up and game over: A systematic literature review of gamification in education ،MDPI .2021 ،

[11] J. P. H. S. G. Meryl Alper ،Interactive technologies for children with special needs ،Ann Arbor, Michigan, USA : Proceedings of the 11th International Conference on Interaction Design and Children .2012 ،

[12] M. Durdyyev ،Mobile Math learning application For children (Fun Math) ، Universiti Teknologi PETRONAS .2012 ،