Architectural Documentation - CodePlay

Table of Contents

Architectural Documentation - CodePlay

1. Introduction

We are a team of four developers working collaboratively to create an educational game using the Godot engine and GDscript programming language. Our project, titled **"Kids Color Book,"** aims to provide an engaging and interactive platform for children aged 5 to 10 to learn fundamental color theory concepts through gameplay.
This document represents our **Architectural Documentation – CodePlay** Report, which systematically captures the functional and non-functional requirements, user needs, and design considerations necessary for the successful development of the game. The report will serve as a reference throughout the development lifecycle, ensuring alignment with the project goals and user expectations. The game's user interface UI is designed in **German/ English** so children can easily play and learn color names in different languages through fun and interactive gameplay. The game also supports **touch input**, allowing children to play easily on tablets with tap and haptic feedback for an engaging experience

1.1. Purpose

This document was created by our development team of four members as part of the planning and design phase for the **"Kids Color Book"** game project. It was compiled through collaborative research, requirement gathering, and analysis sessions, using best practices in software requirement specification.
The purpose of this document is to clearly define the functional and non-functional requirements, user needs, use cases, and design parameters for the game. It serves as a comprehensive guide to align the development team and stakeholders on the project scope, objectives, and constraints.

1.2. Intended Audience

- **Development Team:** To understand, implement, and verify the game features and requirements.
- **Project Managers:** To track progress and ensure alignment with goals.
- **Stakeholders and Sponsors:** To review the project scope and confirm requirements.
- **Testers and QA:** To design tests based on the defined requirements.
- **Target Users:**
- **Children (Ages 5–10):** Primary users who will learn color theory through gameplay.
- **Parents:** Secondary users interested in safe and educational content for their children.
- **Teachers:** Users who may incorporate the game as a learning tool in classroom settings.

1.3. Scope of Use

This document governs the development and implementation of the **"Kids Color Book"** game. It applies to all project phases including design, coding, testing, and deployment. All team members and stakeholders involved in this project are expected to adhere to the guidelines and requirements specified herein to ensure consistency and quality throughout the development lifecycle.

Architectural Documentation - CodePlay

2. Architectural Documentation:

2.1 Discription and Design of System Architecture:

Our color-mixing quiz game is designed for children aged 5–10. We defined clear non-functional requirements to ensure it is accessible, engaging, and robust for young users.

System Type: Desktop-based educational game
Platform: Built using Godot Engine with GDScript
Architecture Style: Modular architecture (game scenes, input logic, feedback system, and UI)

Components:

- **UI Layer:** Displays levels, buttons, color options
- **Game Logic Layer:** Handles color mixing rules, game flow
- **Feedback System:** Manages sounds and animations
- **Score Tracker:** Updates score and level progress
- **Data Storage:** Stores current level, score, and feedback state (in memory; no permanent storage)
- **Restart Handler:** Manages restarting game levels

2.2 Requirements:

Functional Requirements:

- Show 2 base colors per level
- Offer 3 answer choices
- Check answer correctness
- Show visual/audio feedback
- Track score and level

Non-Functional Requirements:

- Runs on low-end desktop systems (min. 2GB RAM, 1 GHz CPU)
- Simple UI with big, bright buttons
- Responsive interaction (< 1 second delay)
- Kid-safe experience (no ads, no external links)

Reference: Section 6 of Requirements Documentation

Architectural Documentation - CodePlay

2.3 Prioritization of Non-Functional Requirements:

Requirement	Priority	Reason
Simple UI	High	Target users (kids) need easy controls
Low System Requirements	High	Must work on classroom/lab PCs
Fast Feedback	Medium	Keeps kids engaged
Educational Focus (No ads)	High	Ensures a safe learning environment

2.4 Architectural Principles:

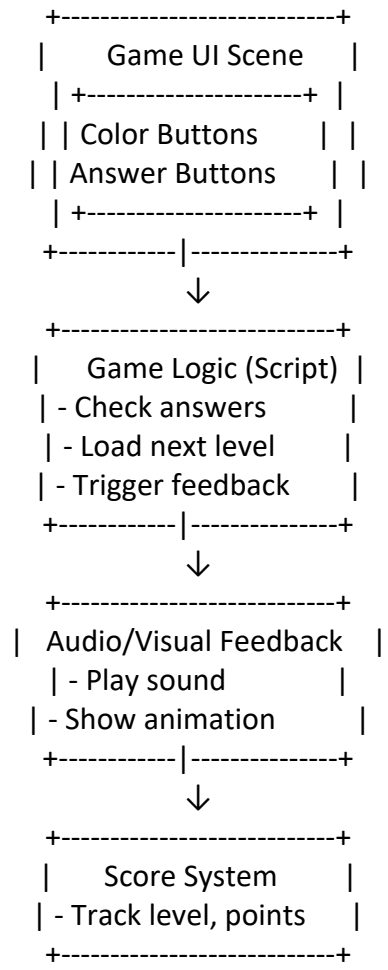
- **Modularity:** Each part (UI, logic, feedback) is separated in scenes/scripts
- **Reusability:** Buttons, animations, and color scenes are reusable
- **Simplicity:** Game avoids complex menus for child accessibility
- **Responsiveness:** Game gives instant feedback to keep attention
- **Consistency:** Same visual style and sounds across all levels

2.5 Interfaces:

Interface	Description
User Interface	Child clicks buttons to mix colors and answer
Audio Feedback Interface	Plays sounds on correct/wrong answers
Visual Feedback Interface	Animations shown after each answer
Level Transition Handler	Starts next level or game over based on result

Architectural Documentation - CodePlay

2.6 System Architecture Diagram:



Activity Flow:

1. Launch app → Start/Game Over Scene.
2. User clicks "Restart Game" → Loads Main Quiz Scene.
3. Game shows current level question with color inputs.
4. User selects one of three answer buttons.
5. If correct → Plays correct sound, updates score, advances to next level.
6. If incorrect → Plays game over sound, navigates back to Game Over screen.
7. Timer enforces 15-second limit per question.
8. Game ends after all levels or on wrong answer.
9. User can restart anytime.

Architectural Documentation - CodePlay

3. System Design:

3.1 System Decomposition:

Module	Responsibility
Game UI	Displays colors, options, buttons, score, and messages
Input Handler	Detects user clicks and button presses
Game Logic	Checks answer correctness, moves to next level or to game over on wrong answer
Feedback System	Plays sounds and animations based on answers
Progress Tracker	Tracks level number and player score
Restart Manager	Restarts game from level 1-5 if the player clicks "Restart Game"

3.2 Design Alternatives and Decisions:

Area	Option A	Option B	Decision + Reason
Game Engine	Unity (C#)	Godot (GDScript)	Godot: lighter and easier for beginners
UI Feedback	Only text messages	Audio + animations	Audio + animations for child-friendly feedback
Answer Checking	Server-side logic	Client-side (in-game) logic	Client-side: offline game
Data Storage	Save to file	Track in memory only	In-memory: sufficient for a short educational game

3.3 Cross-Cutting Concerns:

Concern	How It’s Handled
Usability	Bright visuals, large buttons, voice prompts, minimal text
Performance	Lightweight assets
Accessibility	Voice feedback supports kids who can’t read yet
Error Handling	Shows "Game Over" screen instead of crashing
Security/Privacy	Game runs offline, no personal data is collected
Testability	Each script/module can be tested individually (e.g., answer check logic)

Architectural Documentation - CodePlay

Non-Functional Requirements (Revisited from 2.3)

Requirement	Importance	How It's Supported in Design
Low system requirements	High	Optimized game scenes and asset sizes
Responsive UI	High	Simple interactions and instant feedback
Kid-safe experience	High	No internet, no ads, no data tracking
Easy-to-learn controls	High	Single-click gameplay
Consistent visuals/audio	Medium	Same art and sound style across levels

4. Human-Machine Interface (HMI):

4.1 Requirements for the Human-Machine Interface:

Requirement	Description
Simple and clear visuals	Use large, colorful shapes and buttons children can recognize easily
Voice prompts and feedback	Audio instructions help non-readers (age 5–6) navigate the game
Minimal reading required	Replace text with symbols, icons, and sounds
Single-click interaction	Easy for small children using a mouse or touchscreen
Safe environment	No external links, ads, or in-game purchases
Universal design	Game runs similarly across Windows and macOS desktops

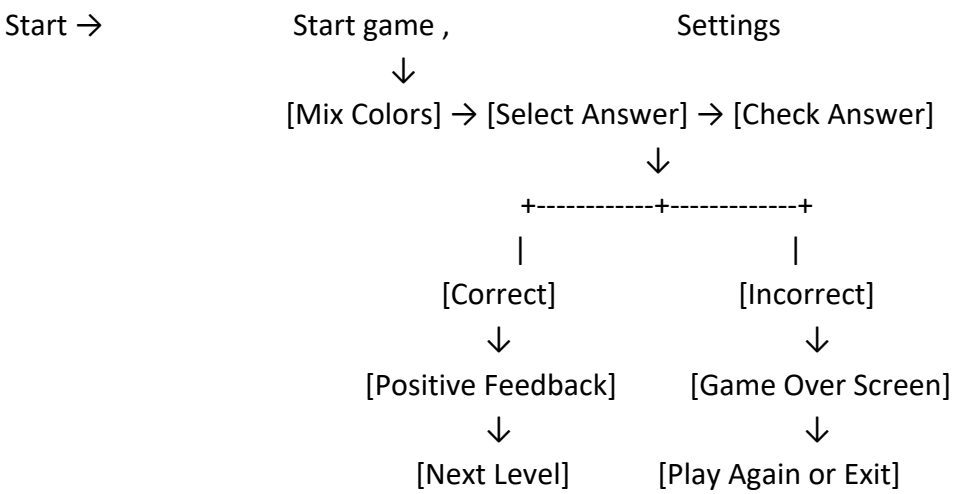
4.2 Design Principles and Style Guide:

Principle	Implementation in the Game
Consistency	Same layout and button style used in all screens
Simplicity	Only essential controls shown (2 base colors, 3 options per level)
Feedback	Bright animation + cheerful sound if correct; gentle error sound if wrong
Affordance	Buttons look clickable (rounded, hover effect); answers clearly distinct
Child-Friendly Aesthetic	Colorful, rounded UI elements; soothing background music

Color Palette: Bright primary and secondary colors
Sound Style: Cheerful xylophone tones and soft voiceovers
Input Style: Single-click or tap interaction

Architectural Documentation - CodePlay

4.3 Interaction Modeling:



Model Ensures:

- Immediate response for each action
- Clear feedback loop (input → result)
- Easy retry path if a child makes a mistake

Architectural Documentation - CodePlay

5. Dictionary/Glossary of Terms:

- **Base Color:**
  A primary color provided as input for mixing in the game (e.g., red, yellow). These colors serve as the starting point for creating new colors.

- **Mixed Color:**
  The resulting color formed by combining two or more base colors (e.g., red + yellow = orange). The player must identify the correct mixed color from answer choices.

- **Level:**
  A stage or round of the game that presents a specific color mixing challenge. Levels typically increase in difficulty or complexity.

- **Feedback:**
  Audio or visual responses given immediately after a player makes a choice. Feedback indicates whether the choice was correct or incorrect and includes animations or sounds.

- **Voice Prompt:**
  Fun and engaging background audio or spoken cues played during gameplay to enhance the player's experience.

- **Game Over:**
  The state reached when a player selects an incorrect answer, ending the current game or level.

- **Restart Game:**
  The action of starting the game again from the beginning, usually triggered by player input after a game over or completion.

- **Restart Timer:**
  A countdown timer that runs after a game over or level completion, giving the player a limited amount of time to decide whether to restart or exit.

- **Score:**
  A numerical value representing the player's progress and success, typically increased by earning points for correct answers.

Architectural Documentation - CodePlay

6. Appendix:

6.1. Tools & Technologies Used

- Game Engine: Godot
- Language: GDScript
- Platform: Desktop (Windows/Linux/Mac)
- Documentation: Microsoft Word or Google Docs

6.2. Project Timeline (Sample)

- Week 1–2: Requirements & Design
- Week 2–6 Game Development
- Week 7: Testing & Feedback
- Week 8: Final Polishing & Submission

6.3. References

- Godot Documentation: https://docs.godotengine.org
- Color Mixing Theory: Basic art education resources from internet.

7. Index

(Refer to Table of Contents at beginning for section navigation.)