Technical Documentation

Project: FunFlip Game

Authors:

Ashish Ghaskata  
Krishna Raj Bhandri

Suraj Bhatta

Mohammad Adnan Khan

**Document name:** AB-Funflip Game-TechnicalDoc

**Creation date:** 20.06.2025

Table of Contents

[1. Introduction 3](#_Toc201706141)

[1.1 Purpose 3](#_Toc201706142)

[1.2 Summary 3](#_Toc201706143)

[1.3 Definitions and Abbreviations 3](#_Toc201706144)

[1.4 References, Standards, and Rules 4](#_Toc201706145)

[2. Core Content 4](#_Toc201706146)

[2.1 System Overview 4](#_Toc201706147)

[2.2 Architecture Design 4](#_Toc201706148)

[2.3 Implementation Details 5](#_Toc201706149)

[2.4 Testing and Quality Assurance 5](#_Toc201706150)

[2.5 Deployment 5](#_Toc201706151)

[2.6 Known Issues and Future Work 5](#_Toc201706152)

[2.7 References to Supporting Artifacts 6](#_Toc201706153)

[2.8. Summary 6](#_Toc201706154)

[2.9. Appendix 6](#_Toc201706155)

Table of Figures

[Figure 1 Activity Diagram 7](#_Toc201706871)

[Figure 2 Use case Diagram 8](#_Toc201706872)

[Figure 3 Sequence Diagram 8](#_Toc201706873)

[Figure 4 High Level System Architecture Diagram 9](#_Toc201706874)

# 1. Introduction

## 1.1 Purpose

This document was collaboratively created by the FunFlip project team as part of the Software Engineering 2 course at TH Aschaffenburg. The content is based on project meetings, requirements analysis, architectural design, and implementation work, and follows the deliverable structure outlined in the course documentation.

**Intended Audience:**

* Project developers and contributors
* Testers and QA engineers
* Supervisors and course instructors
* Future maintainers and stakeholders

**Scope of Use:**  
This document is binding for all team members involved in the design, development, testing, and maintenance of the FunFlip game. It should be referenced throughout the project lifecycle and for any future enhancements or handovers.

## 1.2 Summary

**Scope:**  
FunFlip is an educational 2D card-matching game developed in Godot. The game aims to provide an engaging learning experience through category-based card matching, supporting multiple difficulty levels and user-friendly interactions.

**Stakeholders:**

* Project team (developers, testers)
* Course instructors and supervisors
* End users (students, children)
* Future maintainers

## 1.3 Definitions and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Godot | Open-source game engine used for development |
| GDScript | Scripting language for Godot |
| Scene | A reusable collection of nodes in Godot |
| Asset | Any resource used in the game (e.g., images, sounds, scripts) |
| UI | User Interface |
| QA | Quality Assurance |
| NFR | Non-Functional Requirement |
| SWE | Software Engineering |

**Glossary:**

* **Node:** Basic building block in Godot, representing objects in the scene tree.
* **Sprite:** 2D image or animation integrated into a larger scene.
* **Scene Tree:** Hierarchical structure of nodes in a Godot project.

## 1.4 References, Standards, and Rules

* SWE\_SoSe2025\_DELIVERABLES.pdf (project deliverables and documentation format)
* Godot Engine Documentation (<https://docs.godotengine.org/>)
* Internal project documents in "Project Doc" and "Extra/Doc":
  + Acceptance Documentation SWE.docx
  + Architectural Documentation SWE.docx
  + FunFlip\_Architectural\_Principles.docx
  + FunFlip\_Requirements\_and\_NFR\_Prioritization.docx
  + FunFlip\_System\_Interfaces.docx
* Coding standards for GDScript and Godot projects

# 2. Core Content

## 2.1 System Overview

FunFlip is a 2D educational card-matching game where players select a category (e.g., Animals, Fruits, Vegetables) and a difficulty level. The game presents a grid of cards; players flip cards to find matching pairs. The system tracks moves and score and provides feedback and options to replay or return to the main menu.

**Major Components:**

* **UI Layer:** Start screen, category select, level select, game screen, completion screen
* **Game Logic:** Handles card shuffling, matching logic, move counting, and scoring
* **SceneLoader:** Manages transitions between different screens/scenes
* **Data/Assets:** Sprites, icons, and music organized by category
* **Services:** Audio control, persistent settings (if any)

## 2.2 Architecture Design

**Layered Architecture:**

* **UI Layer:** Godot scenes for user interaction (StartScreen, CategorySelect, LevelSelect, GameScreen)
* **SceneLoader:** Script for managing scene transitions
* **Game Logic Layer:** Main.gd, Game.gd scripts for core logic
* **Data Layer:** Asset folders for images, sounds, and configuration files
* **Services Layer:** Audio management, settings

**Component Diagrams and Relationships:**

* Main.gd manages high-level state and delegates to specific screens
* Game.gd handles gameplay logic and interacts with asset data
* SceneLoader coordinates transitions between UI scenes

**Design Principles Applied:**

* Separation of concerns (UI, logic, data)
* Modularity (scenes and scripts)
* Reusability (category and level selection logic)

## 2.3 Implementation Details

**Technologies and Frameworks:**

* Godot Engine (3.x/4.x)
* GDScript for scripting
* Asset formats: PNG (images), WAV/MP3 (audio)

**Key Classes, Scenes, and Data Structures:**

* **Main.gd:** Entry point, manages scene transitions
* **Game.gd:** Core gameplay logic
* **CategorySelect, LevelSelect:** Scenes for user selection
* **Assets:** Organized by category in folders

**Data Storage and Configuration Files:**

* Card data and categories defined in script or JSON/config files
* Assets stored in structured directories

## 2.4 Testing and Quality Assurance

* Manual playthroughs for all categories and levels
* Test cases for card matching, move counting, and UI transitions
* Review of requirements and NFRs (see FunFlip\_Requirements\_and\_NFR\_Prioritization.docx)
* Bug tracking and issue resolution documented in project notes

## 2.5 Deployment

* Open the project in Godot Engine
* Ensure all assets are in the correct directories
* Run Main.tscn to start the game
* Export instructions for Windows.

## 2.6 Known Issues and Future Work

* Some UI elements may require further polish for accessibility
* Additional categories and levels can be added
* Potential for multiplayer or score persistence in future versions

## 2.7 References to Supporting Artifacts

* Activity and sequence diagrams in "Project Doc/Diagram"
* Requirements and interface documents in "Project Doc"
* Acceptance and architectural documentation in "Project Doc"

## 2.8. Summary

This documentation provides a comprehensive technical reference for the FunFlip game, covering system overview, architecture, implementation, testing, deployment, and supporting artifacts. The modular design and clear separation of concerns facilitate future maintenance and extension.

## 2.9. Appendix

* Glossary of terms and abbreviations
* Asset lists by category
* Additional diagrams (activity, sequence, component)

A diagram of a process

AI-generated content may be incorrect.

Figure 1 Activity Diagram

A diagram of a game

AI-generated content may be incorrect.

Figure 2 Use case Diagram

A diagram of a project

AI-generated content may be incorrect.

Figure 3 Sequence Diagram

A diagram of a software process

AI-generated content may be incorrect.

Figure 4 High Level System Architecture Diagram