# 1. Introduction

## 1.1 Purpose

This document was created by:

* Ashish Ghaskata
* Krishna Raj Bhandari
* Suraj Bhatta
* Mohammad Adnan Khan

It documents the overall project workflow and lessons learned during the development of the **FunFlip Educational Game**.

**Intended audience** includes:

* Project team members
* Supervisors and mentors
* Future maintainers or development teams

**Scope of use**: This document captures the approach, execution, and retrospective learning from the software project. It is useful for quality reviews, audits, and knowledge transfer.

## 1.2 Summary

This document summarizes the engineering practices, methodologies, and lessons learned throughout the development lifecycle of the *FunFlip Game*. It provides insight into how the team applied software engineering concepts, from requirements engineering to testing and handover.

**Stakeholders**: Developers, testers, end users (children aged 4–6), project mentors.

## 1.3 Definitions and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Kanban | Validation and Verification Model |
| HMI | Human-Machine Interface |
| TSCN | Godot Scene File |
| NFR | Non-Functional Requirement |
| QA | Quality Assurance |
| Agile | Iterative development approach focusing on increments |
| SMART | Specific, Measurable, Achievable, Relevant, Time-bound |

## 1.4 References, Standards, and Rules

* FunFlip Game Requirements Document
* Architecture Documentation
* Test Protocols
* ISO 25010 Software Quality Model

## 1.5 Overview

This document includes:

* The structured project approach
* Key activities across the software development lifecycle
* Quality assurance and testing strategy
* Documentation and handover process
* Lessons learned by the team

It is organized into core sections, followed by a reflective summary and appendix if needed.

# 2. Project Approach

## 2.1 Requirements Engineering

* Defined a clear product vision and learning goals
* Used SMART goals, personas, and user stories to reflect user needs
* Documented both:
  + **Functional requirements**: card flip logic, navigation, gameplay
  + **Non-functional requirements**: Quick Response offline operation

## 2.2 System Architecture & Design

* Implemented **5-layer architecture**:  
  UI → SceneLoader → Game Logic → Data → Services
* Ensured **low coupling, high cohesion**, and clean separation of concerns
* Focused on **usability, accessibility**, and **performance**
* Used supporting diagrams: activity flow, domain model, interaction models

## 2.3 Implementation

* Used **Godot Engine 4.x** for development
* Followed **component reuse** (e.g., Card.tscn, AudioControl.gd)
* Applied **data-driven** structure using categories.json
* Adopted **agile-inspired iterations** for incrementally adding features

## 2.4 Quality Assurance & Testing

* Created **detailed test protocols** for functional and non-functional aspects
* Ran **system, integration, and acceptance tests** on Windows Laptop or PC
* Tested under **realistic usage scenarios**
* Included **positive and negative test cases**
* Confirmed **no critical defects** at final acceptance

## 2.5 Documentation & Handover

* Delivered complete documentation:
  + Architecture
  + Requirements
  + Test reports
  + Acceptance documentation
* Validated system's suitability for target users (children aged 4–6)

## 2.6 Process Alignment

* Aligned with **V-Model/W-Model** software process
* Integrated QA from early stages
* Maintained **requirements traceability**
* Applied design principles and focus on usability in every phase

# 3. Lessons Learned

## 3.1 Importance of Early and Clear Requirements

* Clearly defined specs helped reduce confusion later
* Functional and non-functional clarity boosted design efficiency

## 3.2 Value of Modular Architecture

* Layered design improved **reusability** and **maintainability**
* Simplified future updates and bug tracking

## 3.3 Iterative Testing is Essential

* Early unit + integration testing prevented end-phase failures
* Systematic testing cycles improved confidence in product quality

## 3.4 User-Centered Design is Key

* Designing for children required attention to **interface size**, **feedback**, and **simplicity**

## 3.5 Documentation as a Parallel Process

* Keeping docs up to date throughout improved handover
* Saved time and ensured knowledge was not lost

## 3.6 Team Communication

* Clear roles and consistent coordination enabled timely delivery
* Collaboration tools and mutual understanding were critical

## 3.7 Tools and Process Familiarity

* Basic knowledge of **Godot**, **draw.io**, and following structured practices added value
* Sticking to guidelines ensured alignment with academic and engineering standards

**4. Index**

Table of Contents

[1. Introduction 1](#_Toc201799745)

[1.1 Purpose 1](#_Toc201799746)

[1.2 Summary 1](#_Toc201799747)

[1.3 Definitions and Abbreviations 1](#_Toc201799748)

[1.4 References, Standards, and Rules 2](#_Toc201799749)

[1.5 Overview 2](#_Toc201799750)

[2. Project Approach 2](#_Toc201799751)

[2.1 Requirements Engineering 2](#_Toc201799752)

[2.2 System Architecture & Design 2](#_Toc201799753)

[2.3 Implementation 3](#_Toc201799754)

[2.4 Quality Assurance & Testing 3](#_Toc201799755)

[2.5 Documentation & Handover 3](#_Toc201799756)

[2.6 Process Alignment 3](#_Toc201799757)

[3. Lessons Learned 3](#_Toc201799758)

[3.1 Importance of Early and Clear Requirements 3](#_Toc201799759)

[3.2 Value of Modular Architecture 4](#_Toc201799760)

[3.3 Iterative Testing is Essential 4](#_Toc201799761)

[3.4 User-Centered Design is Key 4](#_Toc201799762)

[3.5 Documentation as a Parallel Process 4](#_Toc201799763)

[3.6 Team Communication 4](#_Toc201799764)

[3.7 Tools and Process Familiarity 4](#_Toc201799765)