Software Requirements Specification

for

FUN MATH

Version 1

Prepared by Team 1 of Software Engineering Concepts Class

California State University, San Bernardino

09-10-23

Table of Contents

1. Introduction

1.1 Purpose

The game application "FUN MATH" is an attractive, interactive, and entertaining game. This game is created using the Unity engine. Kids can enhance their mathematical skills by playing this game for fun. The game teaches counting and addition and provides a challenging and engaging educational experience for kids.

1.2 Document Conventions

The document was developed using the IEEE's Software Requirement Specification.

1.3 Intended Audience and Reading Suggestions

The SRS provides a way for the user to verify that the game developed is in line with the original idea. To have a general overview of the project, view the description. Part 2: For a detailed explanation of the gameplay elements and how they connect to the character, see System Features. Part 3: If you are excited about the game's interface and how to use the front-end menus, view External Interface Requirements. Part 4: The technical requirements that the project will hold are listed in Nonfunctional Requirements.

1.4 Product Scope

The game "FUN MATH" aids in teaching kids counting and addition concepts. The various game styles make the concept understandable to players of any skill level. Additionally, this game offers a fun exercise to kids where kids can learn the concept of addition using different items. Hints are also introduced to the users to help them understand the concepts and solve the levels.

1.5 References

- **Software Engineering: A Practitioner's Approach**

  https://ebookcentral.proquest.com/lib/csusb/detail.action?docID=6328275&pq-origsite=primo

- **GitHub page: FUN MATH**

  https://github.com/Harendra1400/Team-Cat

- **Math Kids – Add, Subtract, Count app.**

  Math Kids - Add, Subtract, Count on the App Store (apple.com)

- **NuGet**

  NuGet is a package manager designed to enable developers to share reusable code.

  https://www.nuget.org/

- **ChatGPT**

  https://chat.openai.com/auth/login

2. Overall Description

2.1 Product Perspective

The game is intended to closely imitate the original kid math game app that is available in the Play Store. The primary goal of this game is to make the mathematical concept of addition simpler and more enjoyable for kids to learn.

2.2 Product Functions

Helping kids learn basic addition by implementing addition games of various difficulty levels. Implementing high-quality cartoon animations wherever necessary for the effectiveness of the child's learning and enjoyment. The following is a summary of the major features implemented in the game.

They are separated into categories depending on their function:

- **Title / Menu Screen**: This is the application's initial viewable screen, which includes buttons for play and settings.
- **Cat Character**: A cat character is created to help engage children.
- **Generating random numbers and validating them**: Different numbers are generated along with the correct response for the addition games.
- **Generate questions and validate the answers with reactions**: Adding puzzles are generated and the cat character reacts depending on whether the user gives a correct or incorrect answer.
- **Kid learning progression track**: For every successive puzzle the kid plays, he/she will move to the next level of the game which tracks the development of his/her performance.

2.3 User Classes and Characteristics

The control scheme is designed to be intuitive, and the gameplay is fair enough to be understood by everyone. Therefore, the experience with the game will not be a major factor in dividing the end users. However, any game has some basic division among its end users as hardcore players and casual players. The hardcore players for this game are kids who are attending their primary school.

2.4 Operating Environment

This application will be launched on both the Play Store (Android) and App Store (iOS). A web-based platform is not currently intended.

2.5 Design and Implementation Constraints

The fun math game app for kids is a very minimalistic app both in its functionality and user experience. As of now, there are no constraints that are noticed by the developers.

2.6 User documentation

While this app is a simple and minimalistic one, end users will not face much difficulty.

We have not implemented any complex operations that pose compatibility issues. Kids who are about to start their learning journey can use the app without facing any difficulties.

2.7 Assumptions and Dependencies

We will use Unity 2D for graphics. After the testing phase, we will decide the minimum requirements and the oldest Android version to be supported, then release the game on the market. As of now, we have not noticed any dependencies for this application as the application does not have any complex operations and it is a standalone application.

3. External Interface Requirements

The interface specifications for the system are described in this section of the SRS. User, hardware, software, and communication interface requirements are defined.

3.1 User Interfaces

3.1.1 Main Screen/Home Screen:

This screen is the initial loading screen for the fun math.

Figure 01: Loading Screen/Initial Screen for FUN MATH

3.1.2 Gameplay:

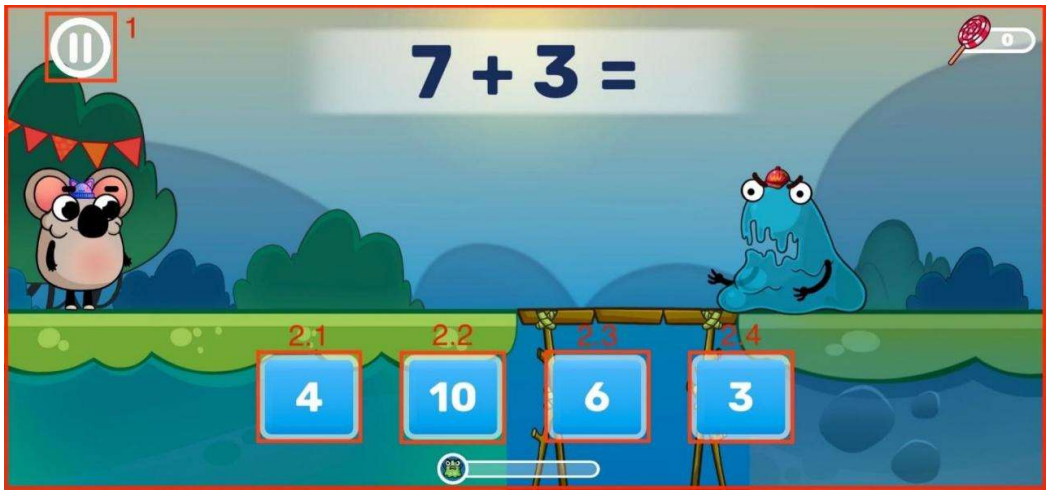

Figure 02: Gameplay Screen

3.1.3 Game over:

Whenever an appropriate task is incomplete, the game over-screen will appear and prompt the user to continue the game.

3.1.3.1 Invoke restart level

When invoked, the current game level restarts

3.1.3.2 Invoke level map window

When invoked, the game is closed and the level map screen appears

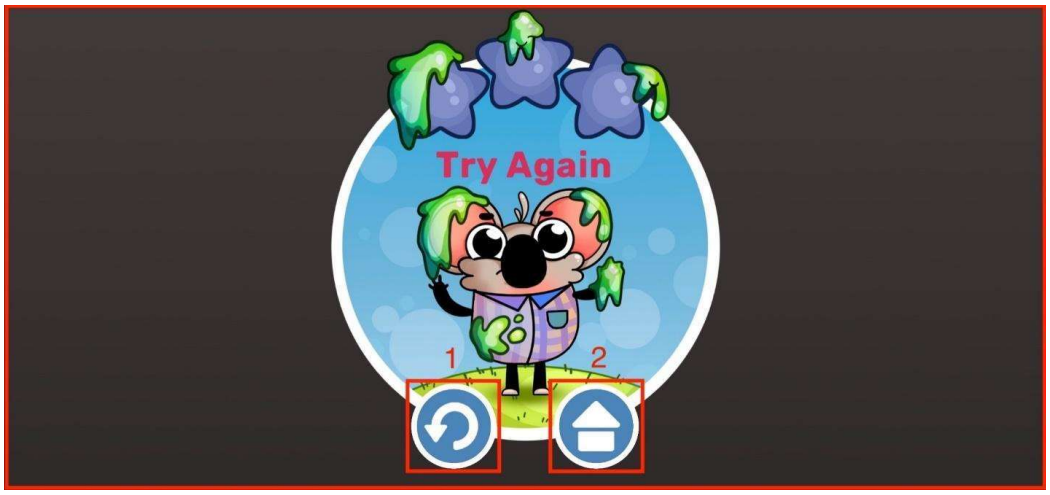

Figure 03: Game over Screen for FUN MATH.

3.1.4. Level Complete:

Figure 04: Level completed Screen for math kid game for kids.

3.1.4.1 Invoke Restart Level

- When invoked, the current game level restarts

3.1.4.2. Invoke Next Level

- When this button is invoked, the next level starts

3.2 Hardware Interfaces

The Math Kid Game for Kids' mobile app may interact with the following hardware interfaces:

3.2.1 Touchscreen Interface

Description: The app relies on the device's touchscreen interface for user interactions, such as tapping buttons, entering answers, and navigating menus.

Functionality: The app should respond to touchscreen gestures, including taps, swipes, and multi-touch interactions.

3.2.2 Speaker and Audio Interface

Description: The app uses the device's speaker and audio interface to provide audio feedback, including sounds, music, and spoken instructions.

Functionality: The app should play audio cues, instructions, and background music through the device's speaker.

3.3 Software Interfaces

3.3.1 Unity

The application will use the Unity game engine for the user interface of the connectome application. While the application will not contain any game components, the framework

makes the interface with the Math Kid Game easy. A 32-bit personal version of the Unity game engine is used for the project. None of the components of the professional version should be needed for the application. Unity version 5.5 is used for the development of the project.

3.3.2 Visual Studio

A streamlined code editor, Visual Studio Code supports development activities like task execution, debugging, and version management. It seeks to offer only the tools a developer needs for a brief code-build-debug cycle and leaves more complicated processes to IDEs with more features, like Visual Studio IDE.

3.4 Communications Interfaces

Since there is no network activity needed, there are no requirements for any interface.

4 System Features

4.1 Title/Menu Screen

4.1.1 Description and Priority

The title screen is the screen the player will see first upon entering the game. Through this interface, the player can choose to start the game or adjust the options. Since the home/menu screen is the hub for all activities in the project, it must be included.

4.1.2 Stimulus/Response Sequences

Step 1: The player will launch the game from their portable device.

Step 2: The start screen loads and appears prompting the player to start the game.

Step 3: The player presses the button and next the actual game begins.

4.1.3 Functional Requirements:

RFQ-1: The home/menu screen must load and appear every time the game is launched.

RFQ-2: If the player quits the game during any stage of a level, they must be returned to the main screen.

RFQ-3: If the player presses the exit button, the game will end.

RFQ-4: If the player completes the game, it will take them to the Level complete screen and then the player can replay the level or go to the next level.

4.2 Generate random numbers and validate the answers

4.2.1 Description and Priority:

The player, after going to the menu screen, navigates to the Adding Puzzle screen, where addition questions are displayed, and the player must choose the correct answer and drag it to the correct location. If the response is correct, an animation of a cat will appear to show that it is correct; otherwise, it prompts the player to let them know that it is an incorrect response. If the answer is correct, it moves on to the next level.

4.2.2 Stimulus/Response Sequences:

Step 1: The player navigates to the gameplay screen from the main screen.

Step 2: On the screen, we can see a cat, and monster, and some random numbers along with accurate numbers for response.

Step 3: An addition question will be generated with random numbers and displayed on the screen.

Step 4: In the answer board random numbers are generated, including the correct answer.

Step 5: Once the answer is selected from the board, a cat animation will pop up acting as to whether the answer is correct or incorrect.

4.2.3 Functional Requirements:

RFQ-1: The cat should appear on the screen and indicate the answer is correct if the player selects the correct answer from the random numbers.

RFQ-2: When the player chooses the correct answer, a cat animation needs to be played.

RFQ-3: When the player chooses the wrong response, it should indicate the answer is incorrect.

RFQ-4: By clicking on the pause menu window button displayed at the top left corner, the player can go back to the screen and choose the game options like back to the home screen or settings.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

Considering the capability of modern smartphones and Android operating systems, performance should not be a problem. Phones with lesser hardware, however, can experience certain issues and may operate slowly. No matter the hardware, the game is designed to provide a fun experience on all Android phones. The game's functioning will be simple enough and easy to understand. The graphics will not be very complex, to avoid slowing down the system.

5.2 Safety Requirement

The Kid Math Game with Cat will not interfere with or harm any of the other applications loaded on the player's phone. Additionally, the game will not make the player's phone overheat, protecting the phone's internal parts from harm. To avoid potential harm to the player, the Kid Math Game with Cat should not be played when the player's attention is split between several things.

5.3 Security Requirements

Since the Kid Math Game with Cat concept won't request any personal information from the player, it cannot compromise such information. This math game can be played without the need for player authentication. Simply downloading the application will allow the gamer to begin playing the additional games. Therefore, this game can be played by anybody who has access to a player's phone. Any unauthorized player will be able to play a math game if they get their hands on the original player's phone. The player must make sure that no other player or individual can access their phone.

5.4 Software Quality Attributes

Kid Math Game concept with Cat responds to the player's directions promptly to guarantee dependability and accuracy. The results of a player's action in the game should be visible to other players in milliseconds, not ten seconds afterward. The Kid Math Game concept with a cat is flexible and adaptable, and it will automatically save the player's progress once each level is finished. In this way, if the player's phone runs out of power while they are playing, they can resume the game at a logical beginning point.

The player will find the graphical user interface to be quite easy to use. The Kid Math Game concept with early levels also gradually introduces all the different commands that the player has access to, as well as any other game features that the player might need to be familiar with to advance. These introductions and pointers won't be provided in more challenging and raised levels because a player will already be familiar with all the resources required to finish the level. This strategy will guarantee that the user experiences a hard game while also gradually learning all the game concepts. Our game does not favor either ease of use or ease of learning more than the other. Any player should be able to pick up and start playing the Kid Math Game idea with Cat right away without spending too much time getting used to the controls. The player won't be overwhelmed by needing to utilize every control at once in the early levels because of the controls' simplicity and intuitiveness. As a player advances through the game and completes each level, the gameplay will gradually introduce each command, making it easier for them to learn and use. The player should be fully capable of using the command introduced in every given level, as well as other commands introduced in earlier levels, by the time that level is through.

5.5 Business Rules

N/A

6 Other Requirements

6.1 Storage Solution

We looked at two potential game data storage systems. Using a file or an integrated database was the option. We chose to utilize Azure App Service as our back-end technology since it offers excellent support for databases and has user-friendly interfaces for getting and storing data. The class Azure App Service provides a simple method to create, edit, and manage Azure App Service. Even if the application were to end abruptly, a database and periodic game data saving will guarantee that no data will be lost.

6.2 Design

The database is created, and the game content is added when the game is installed. The game retrieves the current game state at the beginning, which includes the current task and the path. If there is no current game state, the current task is set to the first task in the newly retrieved route from the database.

6.3 Task

The user's first user interface screen is a welcome screen with a text greeting and a button for moving on to the next activity. The next job in the route is fetched from the database when the user presses the next task button. When the task is finished, the application automatically retrieves the following one. The welcome page will appear once more after the user departs the game, and the route and task are then pulled from the database. The task that the user was working on when the application was terminated will be loaded if the user clicks the next task button.

6.4 Cohesiveness

High cohesion promotes reusability, reliability, and robustness. Each module, class, and file in the entire project is somehow dependent upon one another because it is a gaming project, so it cannot be finished alone. We would need access to both the real character's location as well as the locations of any colliding objects to identify the primary character's collision with any limits, walls, or other obstacles. Because each item in a game must access the information or fields of the other objects for the game's physics to function, practically all game code is cohesive, making this project's cohesiveness higher than that of typical software projects.,?

Appendix: Analysis Models

A. UML Use Case Diagram

Use case diagrams are the diagrams that are used to show the relation between actors and their interactions. A use case diagram shows various use cases and different types of users the system has.

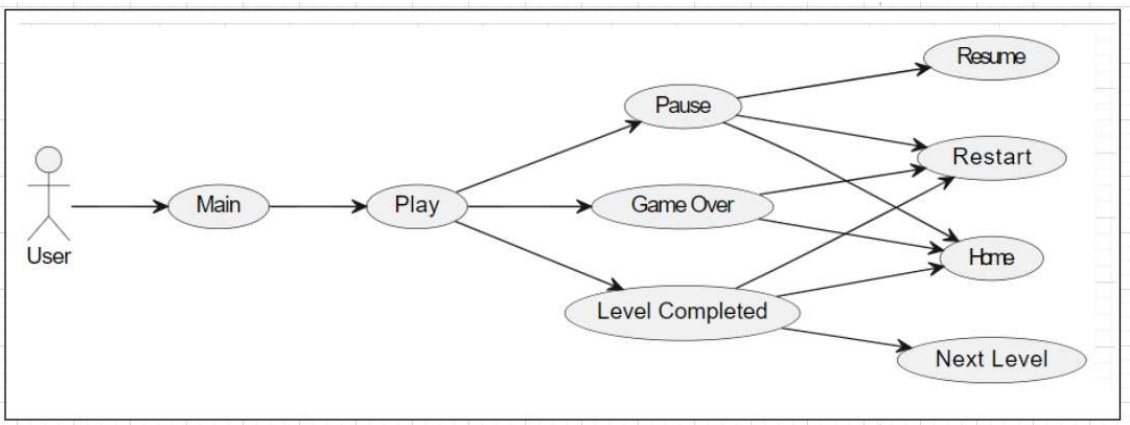

Figure 05: UML Use Case Diagram

B. UML Class Diagram:

UML diagram describes a system by visualizing the different types of objects within a system and the kinds of static relationships that exist among them. It also illustrates the method operations and attributes of the classes.

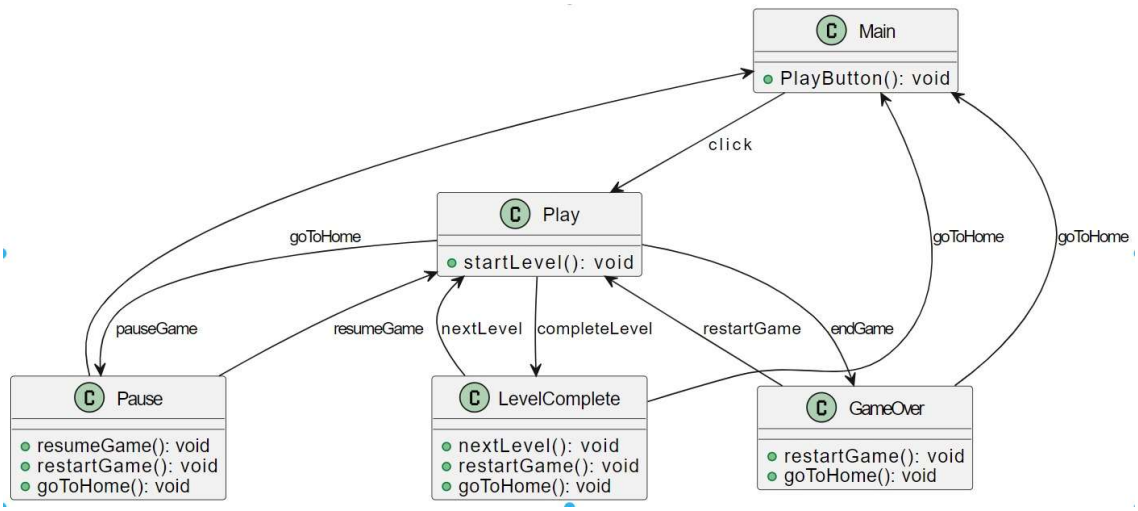

Figure 06: UML Class Diagram

C. UML Sequence Diagram

UML Sequence diagrams illustrate the sequences of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines and the messages that they exchange over time during interaction.

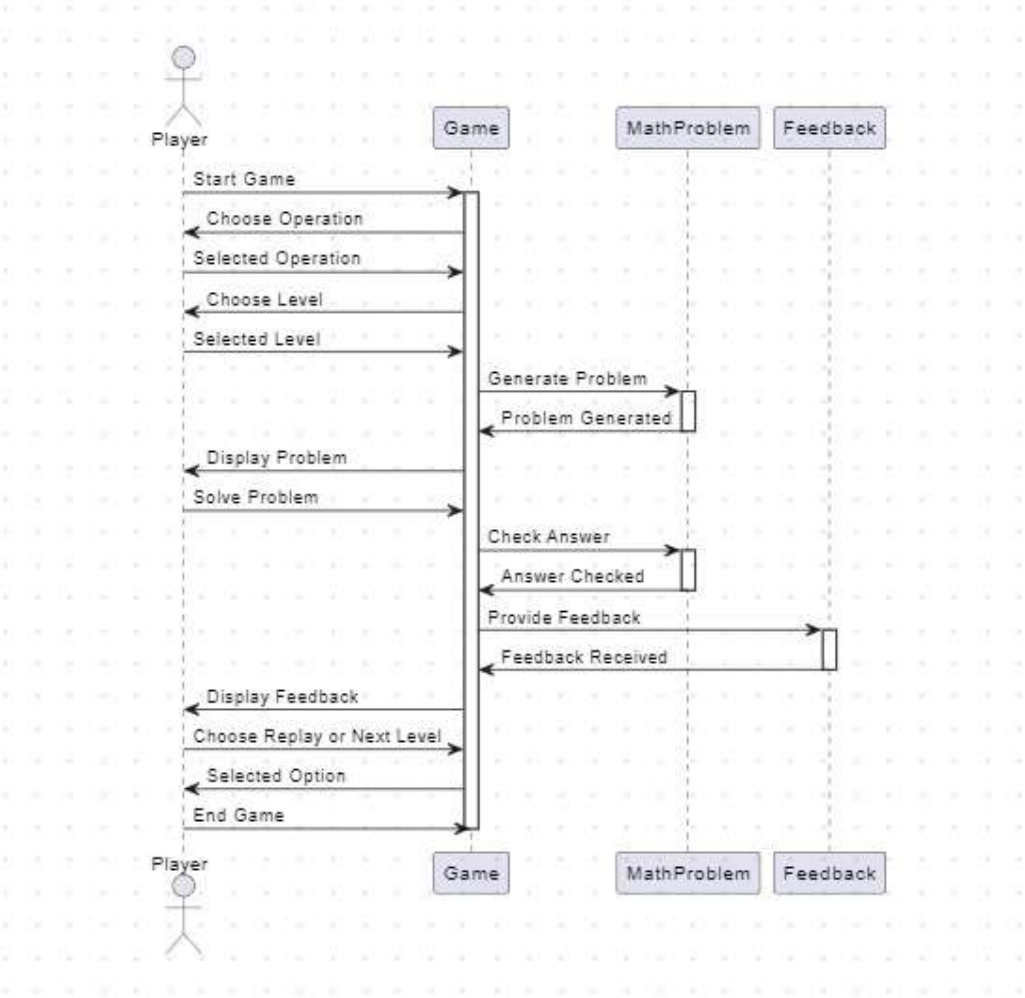

Figure 07: UML Sequence Diagram

Step 1: List of actors

Player: Represents the end-user – initiates the game.

System: Provides the problem to the player.

Database: Confirms the player.

Step 2: Sequence of events

- The player launches the game app.
- The system displays the game menu, allowing the player to select the level.

- The system generates the problem for the player.
- The player submits a solution to the system.
- The system will be sent to the database for verification.
- It sends the confirmation message to the system and the system sends the acknowledgment to the player.
- If the answer is correct, the system generates and displays the next problem. If the answer is incorrect, the player will be allowed to replay.
- The player can choose to reply to the game or the player can select the next level or return to the main menu.

Step 3: Conclusion

- At the end of this sequence, the player has completed a game session and has options to replay the game or choose the next level of the game.