**SRS DOCUMENTATION**

**UPDATES – SUMANA KODURI**

|  |  |
| --- | --- |
| 20 Mar 2023 | 1. Added Safety Requirements 2. System Features 3. Added Interfaces 4. Updated the new Screens |
| 13 Mar 2023 | Updated Performance requirements  Updated class diagram  Updated the new screens |
| 6 Mar 2023 | 1. Updated the content on User Classes and Characteristics 2. Design and Implementation Constraints 3. Added the UML Use Case and Sequence and Class Diagrams 4. Updated the screens like Difficulty Level Screen, Subtraction Puzzle Screen, Subtracting Fun Screen, Subtracting Quiz Screen |
| 27 Feb 2023 | Added Software interfaces   * Unity * Visual Studio * C# and .NET   Updated class, use case and sequence |
| 20 Feb 2023 | * 1. Updated Splash Screen, Beginner Screen   2. Added the content and figures on UML diagrams like Class, Use case and Sequence |
| 13 Feb 2023 | 1. Updated Introduction   * Purpose * Intended audience and reading suggestions * Product scope   2.Updated external interface requirement |
| 6 Feb 2023 | * 1. Created initial version of document   2. Updated SRS with Table of contents |
| 30Jan2023 | 1. Started working on UML Diagrams 2. Started working on Splash screen |

**Software Requirements Specification**

**for**

**Math Games For Kids with Dog**

**Version 1.0**

**Prepared by Team 2 of Software Engineering Class**

**California State University, San Bernardino**

**21.02.2023**

**Table of Contents**

## Table of Contents 1

[**Revision History**](#_yxgx85g0p1v) **2**

[**List of Figures**](#_1ctz5v988vq3) **3**

**1. Introduction 4**

1.1 Purpose 4

1.2 Document Conventions 4

1.3 Intended Audience and Reading Suggestions 4

1.4 Product Scope 4

1.5 References 4

**2. Overall Description 4**

2.1 Product Perspective 4

2.2 Product Functions 5

2.3 User Classes and Characteristics 6

2.4 Operating Environment 6

2.5 Design and Implementation Constraints 6

2.6 User Documentation 6

2.7 Assumptions and Dependencies 5

**3. External Interface Requirements 5**

3.1 User Interfaces 5 3.1.1 Splash Screen 5

3.1.2 [Title/Menu Screen](#_vawen353rv5e) 5

3.1.2 [Subtraction Puzzle Screen](#_am3q721xu5yc) 5

3.1.3 Our Story Screen 5

3.2 Hardware Interfaces 5

3.3 Software Interfaces 6

[3](https://docs.google.com/document/d/112RUbBtUO_gkvtOSvOyUXWlkvBFHb-AF/edit#heading=h.2lwamvv).3.1 C# and .NET 6

[3](https://docs.google.com/document/d/112RUbBtUO_gkvtOSvOyUXWlkvBFHb-AF/edit#heading=h.2lwamvv).3.2 Unity 6

[3](https://docs.google.com/document/d/112RUbBtUO_gkvtOSvOyUXWlkvBFHb-AF/edit#heading=h.2lwamvv).3.3 Visual Studio Code 6

3.4 Communications Interfaces 6

**4. System Features 6**

4.1 Title / Menu Screen 6

4.2 Kid Maths Feature 7

4.3 Our Story Feature 7

4.4 Language Change Feature 7

**5. Other Nonfunctional Requirements 7**

5.1 Performance Requirements 7

5.2 Safety Requirements 7

5.3 Security Requirements 7

5.4 Software Quality Attributes 7

5.5 Business Rules 7

**6. Other Requirements 8**

6.1 [Storage Solution](#_fnx7ko6jvaa0) 8

6.2 [Design](#_ebdhdafmz6t7) 8

6.3 [Task](#_avijltom7o32) 8

6.4 [Cohesiveness](#_l0g7b6hftdy) 8

[**Appendix A: Glossary**](#_qukk4963lw1o) **8**

[**Appendix B: Analysis Models**](#_hwpq8nc87wie) **8**

1. [UML Use Case Diagram](#_23ckvvd) 8

B. UML Class Diagram 8

C. UML Sequence Diagram 8

[**Appendix C: To Be Determined List**](#_7gwf80451ohg) **8**

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Sumana Koduri | 01/30/23 | Created Primary Version of document | 1.0 |
| Sumana Koduri | 02/06/23 | Updated SRS with Table of contents | 1.1 |
| Sumana Koduri | 02/13/23 | 1. Updated Introduction   * Purpose * Intended audience and reading suggestions * Product scope   2.Updated external interface requirement | 1.2 |
| Sumana Koduri | 02/20/23 | 1. Updated Splash Screen, Beginner Screen 2. Added the content and figures on UML diagrams like Class, Use case and Sequence | 1.3 |
| Sumana Koduri | 03/27/23 | Added Software interfaces   * Unity * Visual Studio * C# and .NET   Updated class, use case and sequence | 1.4 |
| Sumana Koduri | 03/06/23 | 1. Updated the content on User Classes and Characteristics 2. Design and Implementation Constraints 3. Added the UML Use Case and Sequence and Class Diagrams   Updated the screens like Difficulty Level Screen, Subtraction Puzzle Screen, Subtracting Fun Screen, Subtracting Quiz Screen | 1.5 |
| Sumana Koduri | 03/13/23 | Updated Performance requirements  Updated class diagram  Updated the new screens | 1.6 |
| Sumana Koduri | 03/20/23 | 1. Added Safety Requirements 2. System Features 3. Added Interfaces 4. Updated the new Screens   And Updated the Class and Sequence Diagrams | 1.7 |

# **List of Figures**

**N/A**

# **Introduction**

## **Purpose**

The game application “Kid Math Game Dog” is an attractive, interactive, and entertaining game. This game is created on the Unity Hub tool. Kids can be able to enhance their mathematical skills by playing this game with fun. It provides learning in counting, subtraction methods. This game provides a challenging and engaging educational experience for kids.

**1.2 Document Conventions**

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

## **Intended Audience and Reading Suggestions**

The Software Requirement Specification provides the way for the user to verify that the game developed is coordinated with the original idea. To completely understand and review the project from initial stage to developed, the content was created in portions and can be therefore read such as described. To have a general overview of the project, view the description Part 2.For a detailed explanation of the game play 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 Part 5.

## **Product Scope**

The game “Kid Math Game with Dog” aids in teaching kids the counting, addition concepts. The mixed math style makes the concept understandable to players of any skill level. Additionally this game offers a fun exercise to kids where kids can be able to learn the concept of addition with different items. Hints are also introduced to the users to make them understand the concept and solve the level in a better way.

## **References**

* **Software Engineering: a Practitioner's Approach**

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

* **GitHub page -**

* **IEEE Template** for System Requirement Specification Documents:

* **Kid Math Game with Dog-**

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

<https://www.nuget.org/>

## **Overall Description**

## **Product Perspective**

The game is an identical replica of 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. This game provides us with the learning experience.

### **Product Functions**

Making the Kids learn basic math calculations by implementing a bingo game for the questions asked based on the difficulty level and thereby pointing towards the answers. Implementing high quality cartoon animations wherever necessary for the effectiveness of the kids' learning. The following is a summary of the major features implemented in the game. This is separated into categories based on those that are necessary for the game to function.

* Title / Menu Screen: This is the application’s initial viewable screen, which includes buttons for a new player, volume control
* Creating characters with names: Different characters are created with names to help us move from one phase to another phase.
* Adjust questions based on difficulty level: There will be a screen where the game redirects us to the next levels based on our performance from easy to medium and hard.
* Generate questions and validate the answers with reactions: Adding puzzles are generated here and the characters created helps us to validate the answers.
* Collecting stamps, stickers, and toys: During the game different toys and stickers can be collected while moving to different levels.
* Kid learning progression track: For every game the kid plays, his/her data is recorded, a certificate is generated which tracks the development of his/her performance.

### **User Classes and Characteristics**

Our app is the simplest and the minimalistic one, end users will not face much difficulty while using the

App as we have not implemented any complex operations on the app that users huge computability.

However, kids who are about to start their learning journey will have great experiences.

### **Operating Environment**

Our application will be launched on both play store(Android) and App store(iOS). As of now, there are no plans for a web-based platform*.*

### **Design and Implementation Constraints**

While our app is the simplest and the minimalistic one, end users will not face much difficulty while using the app as we have not implemented any complex operations on the app that uses huge computability. However, kids who are about to start their learning journey, we will provide them with a tutorial video on how to use the app*.*

### **User Documentation**

While this app is the simplest and the minimalistic one, end users will not face much difficulty. While using the app as we have not implemented any complex operations on the app that uses huge compatibility. However, kids who are about to start their learning journey, we will provide them with a tutorial video on how to use the app.

### **Assumptions and Dependencies**

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

## **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 Splash Screen:** The Splash Screen which will appear whenever we open the game. It appears for 5 seconds and then navigates to the Menu Screen.

Company name

Description automatically generated with medium confidence

**3.1.2 Title/Menu Screen**

This screen consists of subtracting puzzles, subtracting fun, subtracting quiz, and subtracting practice buttons where the kid can be redirected to different scenarios according to his choice.

Graphical user interface

Description automatically generated

* + 1. **Intermediate Screen**

A picture containing text, nature

Description automatically generated

**Graphical user interface

Description automatically generated**

## ​​

## **3.1.2 Subtraction Puzzle Screen**

Graphical user interface

Description automatically generated

**3.1.4 Subtracting Fun Screen**

This screen consists of subtracting numbers using different entries.

A screenshot of a cell phone

Description automatically generated with low confidence

A picture containing text, electronics, display

Description automatically generated

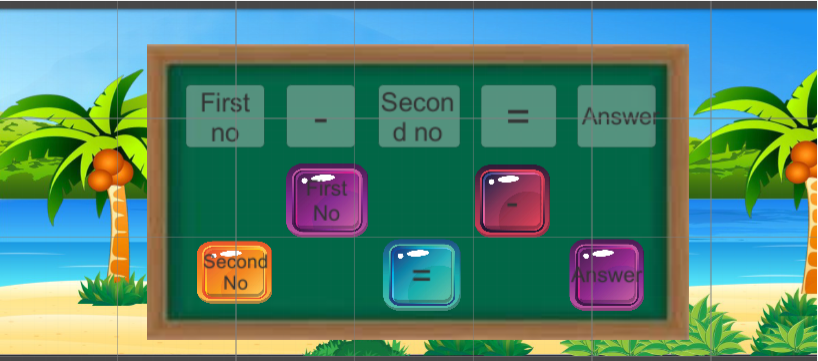
**3.1.5 Subtracting Quiz Screen**

This screen consists of subtracting quiz with different entries.

**Graphical user interface

Description automatically generated**





**3.1.6 Subtracting Practice Screen**

This screen consists of different levels beginner, intermediate and advanced where the kid can be redirected to solve the subtraction practice examples according to his level of progress.

**Graphical user interface, application

Description automatically generated**

**Graphical user interface

Description automatically generated**

## **3.1.9 Our Store Screen**

Wherever clicked on the shop button it redirects to the store screen where the kid can buy a list of items.

### **Hardware Interfaces**

The minimum hardware requirements of Math Kid game are a 500-Megahertz CPU and 1024 megabytes of RAM. Also, a compatible graphics card is required as Math Kid game uses an OpenGL 2D engine to speed up graph visualization. A system with these specifications can handle a network of approximately 1000 edges and nodes.

## **Software Interfaces**

## **C# and .NET**

## **b.** The application will run on version 6.0 of the C# language. This is the most recent version of the C# programming language. The C# programming language is used for all the backend programming for the application. Visual Studio is used to compile all the code that is developed in C#. Version 3.5 of the .NET framework is used for development in C# because of limitations imposed by the Unity game engine.

## 

## **3.3.2 Unity**

## **c.** 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.

## 

## **Visual Studio Code**

## **d.** Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

## 

## **Communications Interface**

**e.** There is no requirement of Communication Interface as there is no Network Activity needed*.*

## System Features

4.1 Title /Menu

**Description and Priority**

The title screen is the screen the player will see every time upon entering the game. Through this interface, the player can choose to start a new game, play from saved data, or adjust the options. Since the title / 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 launched the game from their portable device.

Step 2 : The start screen loads and appears, prompting the player with four buttons “Multiplication Puzzle”, “Multiplication Fun” , “Multiplication Quiz ” and “Multiplication Practice”.

Step 3 : The player clicks the button, triggering to its respective function.

**4.1.3 Functional Requirements**

RFQ-1 : The title / 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 retired to the menu screen.

RFQ-3 : If the player presses the exit button, the game will end and return the player to the phone’s regular interface.

**Step 1** : The player launched the game from their portable device.

**Step 2** : The start screen loads and appears, prompting the player with two buttons “New Player” and “Report Card”.

**Step 3** : The player presses the button, triggering its respective function.

* + 1. **Functional Requirements**

**RFQ-1** : The title / 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 retired to the menu screen.

**RFQ-3 :**If the player presses the exit button, the game will end and return the player to the phone’s regular interface.

### **Creating Multiple Players Names**

* + 1. **Description and Priority**

The player opens the game from their Android or iOS device and presses the “New Player” button to create multiple player names up to 10 using alphabets or numbers.

**4.2.1 Description and Priority**

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.2.2 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate, when necessary, information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

* + 1. **Stimulus/Response Sequences**

**Step 1** : The player presses the “New Player” button on the game-play interface.

**Step 2 :** A popup screen appears to enter the player name using the in screen keyboard.

**Step 3** : The player presses the submit button, triggering its respective function to the next screen.

* + 1. **Functional Requirements**

**RFQ-1** : The progress of each player based on the player name will be saved in the azure app backend services.

**RFQ-2** : The players can change the game level anytime during the game by returning to the “Menu/ Title page” and selecting the different player name.

### Adjust Questions based on Difficulty Level

* + 1. **Description and Priority**

Once the player's name has been input, it will be navigated to a percentage concept screen with a popup screen with three buttons—"Easy," "Medium," and "Hard"-may be viewed. The player can use these buttons to select which level of the game the player wants to enter. Every user can play any level based on their choice.

* + 1. **Stimulus/Response Sequences**

**Step 1** : The player enters the name and presses the “Submit” button to navigate the next screen.

**Step 2** :In this next screen the player will press the “percentage concept button”.

**Step 3** : Following this a popup with three buttons will appear as "Easy," "Medium," and "Hard" and the player can select his desired screen.

**Step 4** : The player presses any of the three buttons, triggering its respective function to the next screen*.*

* + 1. **Functional Requirements**

**RFQ-1** : The questions will range 0-20 in easy level and 20-50 in medium level and 50-100 in hard level respectively.

### Teacher Hints to Understand Concepts

* + 1. **Description and Priority**

If a player is having trouble answering a question on the bingo screen, they can use the hints button at the bottom of the screen, which provides a clue for each question, allowing them to go on to the next one and solve the problem. We must begin counting the number of dots in these hints to determine the solution.

* + 1. **Stimulus/Response Sequences**

**Step 1** : The player first sees the question and tries to answer.

**Step 2** : If the player needs help for any questions the player can click on an egg shaped question mark visible on the screen.

**Step 3** : The Hints screen will pop up and the player can count the number of dots appearing on the screen this will help the player to provide the correct answer.

* + 1. **Functional Requirements**

**RFQ-1** : On the bingo screen, the Hints button must be visible for every question of the easy, medium, and hard levels.

**RFQ-2** : There should be a number of dots or some visual cue in the Hints screen so that the player may verify the right response.

### **Our Story Feature**

* + 1. **Description and Priority**
    2. **Stimulus/Response Sequences**
    3. **Functional Requirements**

### **Language Change Feature**

* + 1. **Description and Priority**

# **Other Nonfunctional Requirements**

### **Performance Requirements**

Considering the capability of modern smartphones and android operating systems, performance shouldn’t 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 functioning will be simple enough and easy to understand. The graphics won't be extremely complex to avoid slowing down the system.

### **Safety Requirements**

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.

### **Security Requirements**

### **Software Quality Attributes**

### **Business Rules**

N/A

## **Other Requirements**

### **Storage Solution**

### **Design**

### **Task**

### **Cohesiveness**

## **Appendix A: Glossary**

## **Appendix B: Analysis Models**

## **A.UML Use Case Diagram:**

Use cases diagrams are the diagrams which 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 and will often be accompanied by other types of diagrams as well.

Diagram

Description automatically generated

## **B.UML Class Diagram:**

UML diagram type that 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 operations and attributes of the classes.

MATH KID

|  |
| --- |
| Main Screen |
| Counting Adding Puzzle Adding Fun Adding Quiz  Addition Practice Compare Subtracting Puzzle Subtracting Fun Subtracting Quiz Subtracting Practice |
|  |

MATH KID

|  |
| --- |
| Main Screen |
| Counting Adding Puzzle Adding Fun Adding Quiz  Addition Practice Compare Subtracting Puzzle Subtracting Fun Subtracting Quiz Subtracting Practice |
|  |

|  |
| --- |
| Game Screen |
| Counting Scene Adding Puzzle Scene Adding Fun Scene Adding Quiz Scene  Addition Practice Scene Compare Scene Subtracting Puzzle Scene Subtracting Fun Scene Subtracting Quiz Scene Subtracting Practice Scene |
|  |

Corresponding

|  |
| --- |
| KIDS MATH |
| Numbers Signs |
| Subtracting Puzzle Subtracting Fun Subtracting Quiz Subtraction Practice |

Random functions () Quiz ()

Numbers Average Mark

Subtracting Quiz

|  |
| --- |
| Subtracting Puzzle |
| Numbers Signs |
| Random functions () Puzzle () |

|  |
| --- |
|  |
|  |
|  |

|  |  |  |
| --- | --- | --- |
| Subtracting Fun  Numbers Average Mark  Random functions() Fun() | | |
|  | |  |
|  | Subtraction Practice | |
| Numbers Average Mark | |
| Random functions() Practice() | |

|  |
| --- |
| **Home Screen** |
| logoImage: dog Image subtractionPuzzleButton: Button subtractingFunButton: Button subtractingQuizButton: Button  subtractingPracticeButton: Button |
| display(): void  launchSubtractionPuzzle(): Random Function() launchSubtractingFun(): Random Function() launchSubtractingQuiz(): Random Function()  launchSubtractingPractice(): Random Function() |

uses-a

Load():

**Image**

Uses-a

**Button**

Click():

**Subtracting Puzzle**

num1: int num2: int answer: int operand: char

SubtractingPuzzle() generatePuzzle():

checkAnswer(userAnswer:int):boolean

|  |  |  |  |
| --- | --- | --- | --- |
| **Subtracting Fun** |  | **Question** | |
| Operands: int[] Operators: char[] Answer: int | |
| Level: int User: int Time: int User: user |
| startGame(): endgame(): nextLevel(): displayQuestion(): |
|  | |  |
|  | | |

name: String age: int gender: String

**User**

|  |  |  |
| --- | --- | --- |
| **SubtractingQuiz** | | |
| Difficulty  nonQuestions currentScore questions | | |
| startQuiz ()  submitAnswer ()  getNextQuestion () endQuiz () | | |
|  | |  |
|  | **Question** | |
|  | operand1 operand2  answer | |
|  | getQuestionText ()  checkAnswer () | |

|  |  |  |
| --- | --- | --- |
|  | **SubtractingBeginner** | |
| numQuestions | |
| currentQuestion | |
| currentScore | |
| questions | |
| startPractice() | |
| submitAnswer() | |
| getNextQuestion() | |
|  | |  |
| operand1 operand2 answer | | |
| getQuestionText() checkAnswer() | | |

has-a

difficulty: string numQuestions: int numCorrect: int

questions: List<SubtractionQuestion> display(): void

generateQuestions(): void chechAnswer(questionNum, answer): Boolean getScore(): String

**Subtracting Practice Intermediate**

|  |
| --- |
| **SubtractingQuestion** |
| minuend: subtrahead: answer: |
| display(): |

generateQuestion(): checkAnswer(): boolean

Difficulty: String Num1: Int Num2: Int Answer: Int UserInput: Int

Subtracting Practice

Settings Practice

**Math Kid Mobile Application**

### 

### **C.UML Sequence Diagram:**

UML Sequence diagram that illustrates the sequence 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 the interaction.

Diagram

Description automatically generated with medium confidence

### **Appendix C: To Be Determined List**

**N/A**