**SRS DOCUMENTATION**

**UPDATES – SUMANA KODURI**

|  |  |
| --- | --- |
| 10 April 2023 | Updated  Stimulus/Response  Functional Requirements  Software Quality attributes  Business Rules  Updated New screens |
| 03 April 2023 | Updated   1. System Features 2. Description and Priorities 3. External Interface Requirements 4. Other Non-Functional Requirements 5. Updated UML Use case, class, sequence diagrams 6. Updated new screens |
| 20 Mar 2023 | 1. Added Safety Requirements 2. Added Interfaces 3. 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 ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 4

3.1.1 Initial Screen 4

3.1.2 Main Screen 4

3.1.3 Subtracting Puzzle Screen 4

3.1.4 Subtracting Fun Screen 5

3.1.5 Subtracting Quiz Screen 5

3.1.6 Subtracting Practice Screen 5

3.1.7 Store Screen 5

3.2 Hardware Interfaces 5

3.3 Software Interfaces 5

3.4Communications Interfaces 5

**4. System Features 5**

4.1 Title/Menu Screen 6

4.1.1 Description and Priorities 6

4.1.2 Stimulus/Response 6

4.1.3 Functional Requirements 7

4.2 Generate random numbers and validate the answers 7

4.2.1 Description and Priority 7

4.2.2 Stimulus/Response 8

4.2.3 Functional Requirements 8

4.3 Subtracting Quiz Screen 8

4.3.1 Description and Priority 9

4.3.2 Stimulus/Response 9

4.3.3 Functional Requirements 9

4.4 Subtracting Puzzle Screen 9

4.4.1 Description and Priority 9

4.2.2 Stimulus/Response 9

4.3.3 Functional Requirements 10

**5. Other Nonfunctional Requirements 11**

5.1 Performance Requirements 11

5.2 Safety Requirements 11

5.3 Security Requirements 11

5.4 Software Quality Attributes 11

5.5 Business Rules 12

**6. Other Requirements 12**

**Appendix A: Glossary 13**

**Appendix B: Analysis Models 14**

1. UML Use Case Diagram 15
2. UML Class Diagram 16
3. UML Sequence Diagram 17

**Appendix C: To Be Determined List**

# **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 ScreensAnd Updated the Class and Sequence Diagrams | 1.7 |
| Sumana Koduri | 04/03/23 | Updated   1. System Features 2. Description and Priorities 3. Stimulus/Response 4. Functional Requirements 5. External Interface Requirements 6. Other Non-Functional Requirements 7. Updated UML Use case, class, sequence diagrams 8. Updated new screens | 1.8 |
| Sumana Koduri | 04/10/2023 | Updated  Stimulus/Response  Functional Requirements  Software Quality attributes  Business Rules  Updated New screens | 1.9 |

# **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

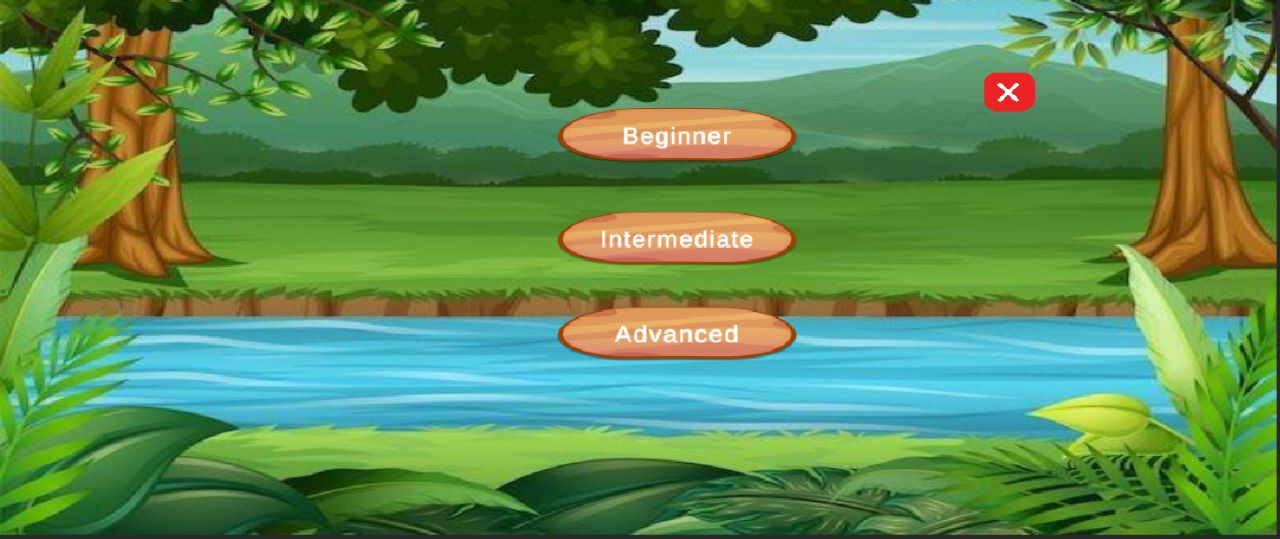
Description automatically generated with medium confidence

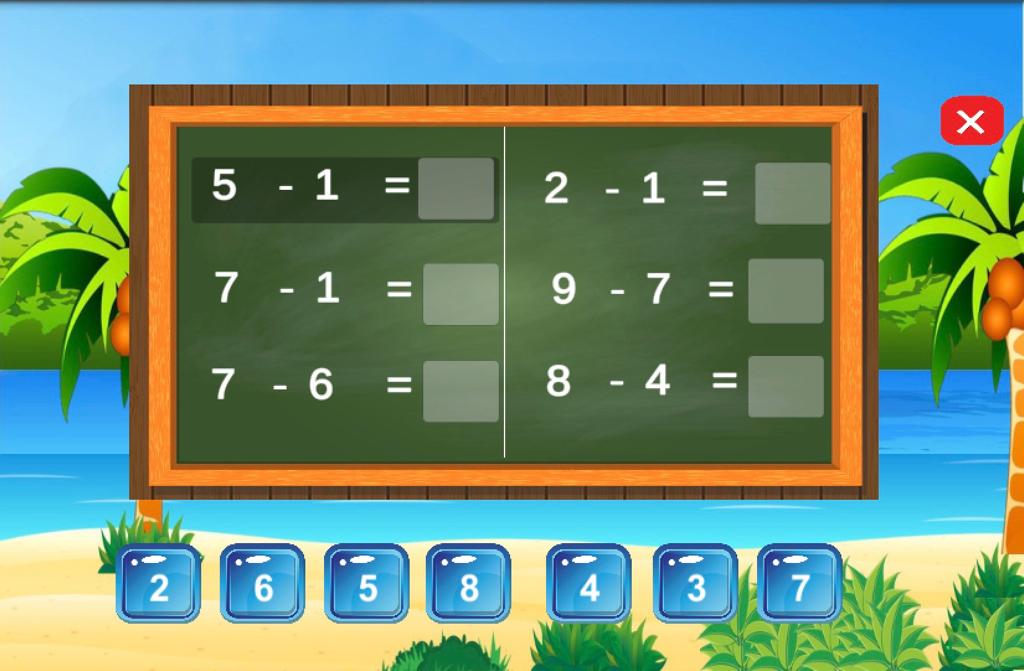
**3.1.2 Title/Menu Screen**

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.

****

* + 1. **Intermediate Screen**

****



## ​​

## **3.1.4 Subtraction Puzzle Screen**

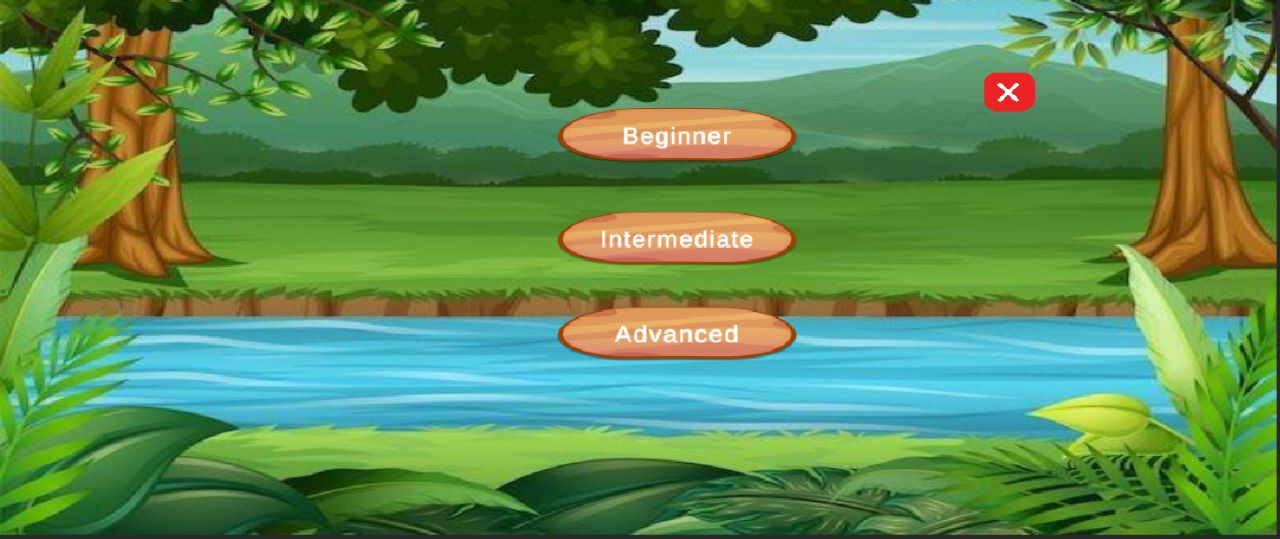
This screen consists of solving the puzzles with different numbers and addition operators.

Graphical user interface

Description automatically generated

**3.1.5 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

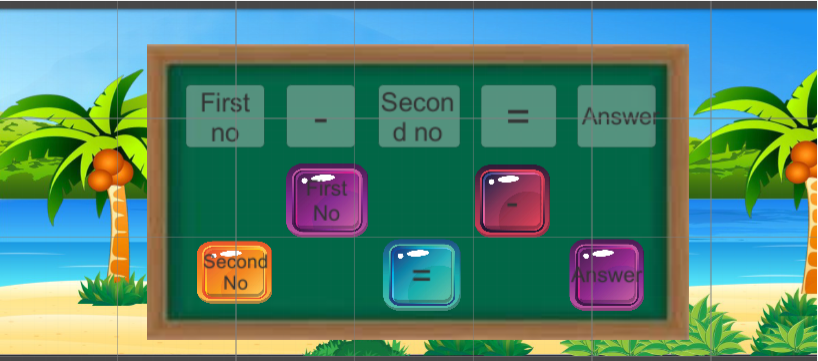
**3.1.6 Subtracting Quiz Screen**

This screen consists of subtracting quiz with different numbers.



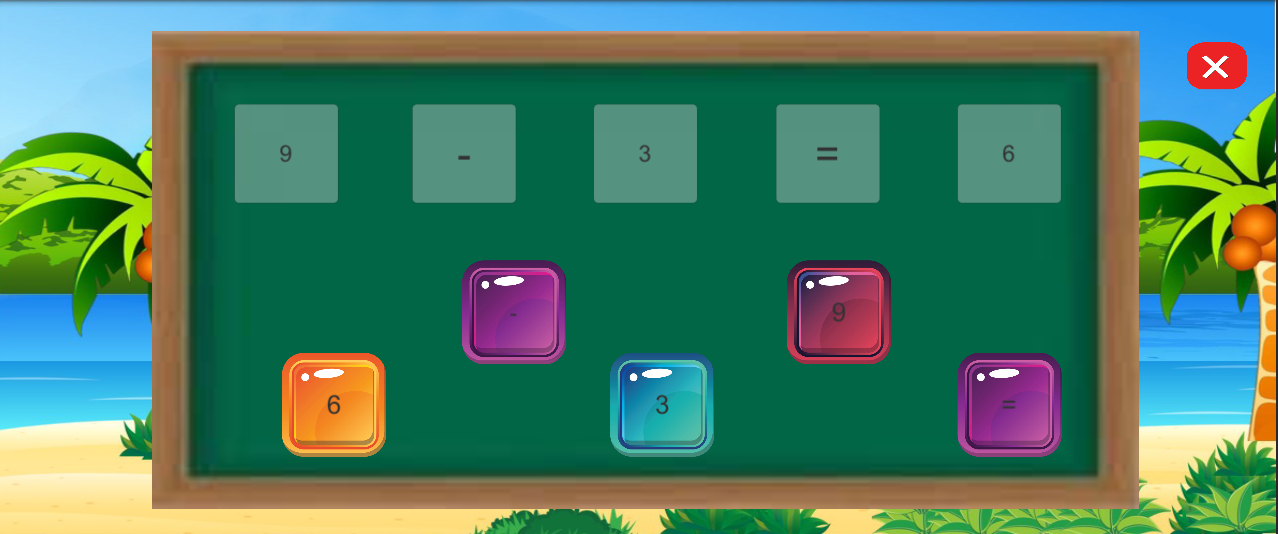






**3.1.7 Subtracting Practice Screen**

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**

Graphical user interface

Description automatically generated

## 

## 

## **3.1.8 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

## 4.1 Title/Menu Screen

## 4.1.1 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 the game, play the resumed game, 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 will launch the game from their portable device.

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

game

Step 3 : The player presses the button, triggering to different functionalities of the

game, whether it could be adding puzzle, adding quiz, adding fun and

addition practice.

## 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 main screen.

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

player to the mobile’s regular interface.

RFQ-4 : If the player completes the game, it will take to the rewards page and

then returns the player to the menu screen.

## 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 questions regarding the addition practice are displayed in the questionnaire

board and here the player must choose an answer and drag the appropriate answer

to the correct place.If the response is accurate, an animation of a cat will appear to

show that it is accurate; otherwise it prompts the player to let them know that it is a

wrong response.If the answer is accurate it moves to the next level.

## 4.2.2 Stimulus/Response Sequences :

Step 1 : The player navigates to the adding puzzle screen from the main menu

screen.

Step 2 : In the adding puzzle screen we can see a questionnaire board and some

random numbers along with accurate number for response.

Step 3 : An addition question will be generated with random numbers and

displayed on the questionnaire board.

Step 4 : In the answer board random numbers are generated to select the correct

answer.

Step 5 : Once the answer is selected from the board, it will either popup a cat

animation if it is correct or displays as incorrect if it is wrong.

## 4.2.3 Functional Requirements :

RFQ-1 : The cat should appear and prompt the answer as correct on the screen if the

player selects the correct answer from the random numbers.

RFQ-2 : When the player chooses the correct answer and drops it in the right place,

a cat animation needs to play.

RFQ-3 : When the player chooses the wrong response, it should prompt as incorrect

answer.

RFQ-4 : By clicking on the home button displayed at the top right corner, the player

can go back to the menu screen and start the game again.

## 4.3 Subtracting Quiz Screen

## 4.3.1 Description and Priority :

The player after going to the menu screen navigates to the adding quiz screen

where questions regarding the addition practice are displayed in the questionnaire

board and here the player must choose an answer and drag the appropriate answer

to the correct place.If the response is accurate, an animation of a cat will appear to

show that it is accurate; otherwise it prompts the player to let them know that it is a

wrong response.If the answer is accurate it moves to the next level.

## 4.3.2 Stimulus/Response Sequences :

Step 1 : The player navigates to the adding quiz screen from the main menu

screen.

Step 2 : In the adding quiz screen we can see a questionnaire board and some

random numbers along with accurate number for response.

Step 3 : An addition question will be generated with random numbers and

displayed on the questionnaire board.

Step 4 : In the answer board random numbers are generated to select the correct

answer.

Step 5 : Once the answer is selected from the board, it will either popup a cat

animation if it is correct or displays as incorrect if it is wrong.

## 4.3.3 Functional Requirements :

RFQ-1 : The cat should appear and prompt the answer as correct on the screen if the

player selects the correct answer from the random numbers.

RFQ-2: When the player chooses the correct answer and drops it in the right place,

a cat animation needs to play.

RFQ-3: When the player chooses the wrong response, it should prompt as incorrect

answer.

RFQ-4: By clicking on the home button displayed at the top right corner, the player

can go back to the menu screen and start the game again.

## 4.4 Subtracting Puzzle Screen

## 4.4.1 Description and Priority:

The player after going to the menu screen navigates to the adding puzzle screen by

clicking on the adding puzzle button where some puzzles with numbers, addition

operators and equal to operators regarding the addition practice are displayed in the

questionnaire board and here the player must choose an answer and drag the

appropriate number and operator to the correct place. If the response is accurate, an

animation of a cat will appear to show that it is accurate; otherwise, it prompts the

player to let them know that it is a wrong response. If the answer is accurate it moves

## 4.4.2 Stimulus/Response Sequences:

Step 1: The player navigates to the adding puzzle screen from the main menu

screen.

Step 2: In the adding puzzle screen we can see a questionnaire board and some

random numbers along with addition and equal to operators to complete the puzzle.

Step 3: An addition puzzle will be generated with numbers and addition, equal to

operators and displayed on the questionnaire board.

Step 4: In the answer board random numbers are generated along with + and =

operators to drag them to the correct place to form a perfect addition operation.

Step 5: Once the answer is selected and dropped in an appropriate place from the

board, it will either popup a cat animation if it is correct or displays as incorrect if

it is wrong.

## 4.4.3 Functional Requirements :

RFQ-1: The cat should appear and prompt the answer as correct on the screen if the

player selects and drops the correct number and operator from the generated puzzle.

RFQ-2: When the player chooses the correct answer and drops it in the right place,

a cat animation needs to play.

RFQ-3: When the player chooses the wrong response, it should prompt as incorrect

answer.

RFQ-4: By clicking on the home button displayed at the top right corner, the player

can go back to the menu screen and start the game again.

### 

# **Other Nonfunctional Requirements**

### **5.1 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.

### **5.2 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**

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

### **Software Quality Attributes**

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

### 

### **Business Rules**

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

N/A

## **Other Requirements**

### **Storage Solution**

We looked at two potential game data storage systems. Using a file or an integrated database were the options. 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. With the class Azure App Service, it provides a simple method to create, edit, and manage Azure App Service. Additionally, only the parent application may see the content of each database that is created. Even if the application were to end abruptly, a database and periodic game data saving will guarantee that no data will be lost.

### **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.

### **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.

### **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 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.

Table

Description automatically generated with low confidence

Diagram

Description automatically generated

Diagram

Description automatically generated

**Home Screen**

-logoImage: dog Image

-subtractionPuzzleButton: Button

-subtractingFunButton: Button

-subtractingQuizButton: Button

-subtractingPracticeButton: Button

+display (): void

+launchSubtractionPuzzle (): int

+launchSubtractingFun (): int

+launchSubtractingQuiz (): int

+launchSubtractingPractice (): int

uses-a

**Image**

Load ():

Uses-a

**Button**

Click ():

|  |
| --- |
| **Subtracting Puzzle**  num1: int  num2: int  answer: int  operand: char    Subtracting Puzzle ()  generate Puzzle ():Void  checkAnswer(userAnswer:int): Boolean |

Diagram

Description automatically generated

**Subtracting Quiz**

-difficulty

-nonquestion’s

-current Score

-questions

**+**start Quiz ()

+submit Answer ()

+getNextQuestion ()

+ end Quiz ()

**Question**

-operand1

-operand2

-answer

+getQuestionText ()

+checkAnswer ()

**Subtracting Beginner**

+getQuestionText()

+checkAnswer()

-nonquestion’s

-current Question

-current Score

-questions

+start Practice ()

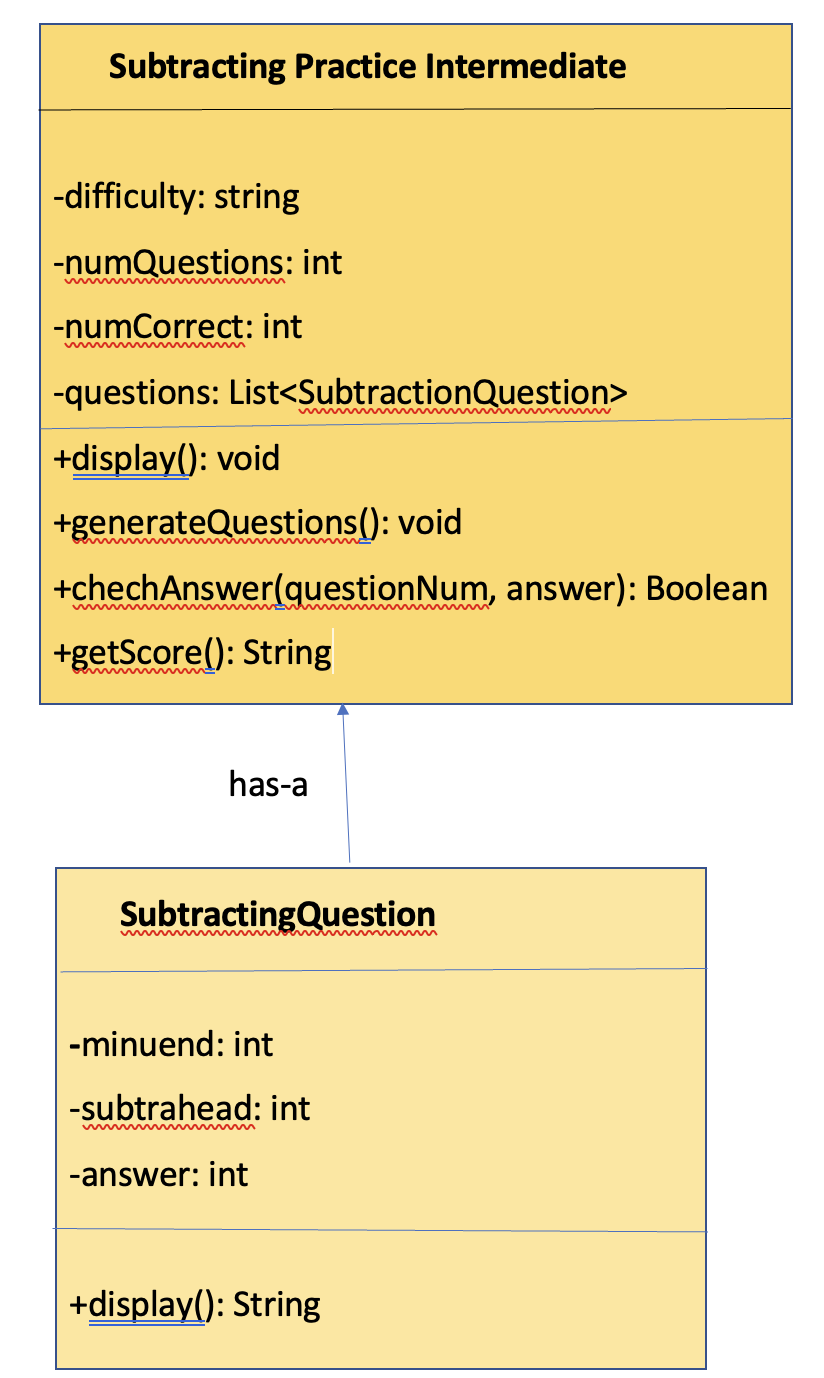
+submit answer()

+getNextQuestion ()

-operand1

-operand2

-answer



### 

### **A picture containing table Description automatically generated**

### **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

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

**N/A**