TMT

Taste Mongolian Tradition

Software Engineering

Project Report

Submitted by:
Name (Roll no.)

Supervisor:

And Munkhzul 3500

Sai Kiran 3407

Department of Computer Science
Hansraj collage

University of Delhi

Problem Statement

Culture refers to the beliefs, values, practices, customs, and social behaviors that define a society. It encompasses language, religion, food, music, art, and much more. Understanding various cultures is important in today's globalized world as it helps in breaking down cultural barriers and promoting cross-cultural understanding and respect. Knowing about different cultures can improve communication and foster relationships in personal, professional, and international contexts. Additionally, knowledge about other cultures can broaden one's perspective and increase cultural intelligence, making one more adaptable to diverse social situations. Understanding other cultures can also help in avoiding misunderstandings and promote peaceful relationships, especially in a world where people of different cultures interact daily. In conclusion, being familiar with various cultures can lead to better relationships, greater cultural competency, and a more tolerant and harmonious world.

This project aims to introduce the culture of the largest empire in history by implementing a pc game. Mongolian traditional ankle bone games as an important intellectual cultural heritage, perfect in terms

of playing methods and content based on the life and livelihood of nomads. Those games/shagain naadgai are unique in that it teaches children about hard work and morals, such as raising livestock, hunting, and showing respect to brothers and sisters.

Additionally by working on this project we will have basic knowledge to make pc games.

Platform to use:

- UNITY

This game engine has a great community, a low entrance level, and a smooth learning curve. But at the same time, it is pretty strict and has many powerful features to write stable and effective code. All this makes Unity technologies perfect for beginners and mature programmers.

Known projects done in unity

- 3D chess(2020)
- City maker(2021)

Programming Language to use:

- C++, C#

The C++ programming language is compatible with popular gaming engines like Unity . Frequent development updates keep C++ aligned with modern gaming requirements.

Process Model

We will be following Water Fall model, as it is a linear and sequential approach to software development that consists of several phases that must be completed in order & the project is not complex and requirements are well known. Each phase must be completed before the next phase can begin. The requirements for the software system are gathered from stakeholders' software requirements are documented and reviewed by the project team. Based on the requirements gathered, the architecture, interfaces, and other design elements are created. On next phase the software system is developed based on the design created in the previous phase. This is the phase where the actual coding takes place. fourth phase of the waterfall model is the testing phase, in which the software system is tested for defects and bugs. During this phase, the system is tested at different levels to ensure that it meets the requirements and functions correctly. final phase of

the waterfall model is the deployment phase, in which the software system is deployed to the production environment .

Software Requirement Specification

Overall Description

Product Functions

The game will include only two parts:

Fortune telling: Allows players to toss the coins together and obtain a fortune on the resulting combinations.

Racing part: Allows players to move their game pieces based on how many horse faces they can toss.

User Characteristics

The game designed for players from all age groups and they can have fun and enjoy the whole features.

General Constraints

The game must follow and make feel of the great Mongolian culture through entire part.

Assumptions and Dependencies

Assumptions

It is assumed that players will have a basic understanding of the rules of the TMT game and how to plays it and have the target audience for the software will have access to the necessary hardware and software to run the game, such as a smartphone or computer.

Dependencies

The software may depend on compliance with local laws and regulations, such as data privacy laws or gambling regulations.

External Interface Requirements

User Interfaces

The user interfaces should be designed to be easy to use and eyes catching, with clear information and simple workflows.

Hardware Interfaces

The Game should be compatible with different types of hardware, such as personal computers and laptops with different windows versions.

Software Interfaces

the game may need to interface with external software systems. Such as game music file selection must be connect with files.

Functional Requirements

Main Menu

1.The software should provide access to all modules of the game from the main menu.

2.The software should allow the user to start a new game or resume an existing game.

Fortune-Telling Module

1.The software should generate random combinations of coin faces to determine the user's fortune .

2.The software should display the fortune to the user in a clear and understandable way.

3.The software should allow the user to toss the coins multiple times to receive different fortunes.

Racing Module

1.The software should generate a game board that displays the position of each player.

2.The software should allow players to toss the four coins to determine how many spaces they can move on the game board.

3.The software should display the outcome of each turn and update the position of each player accordingly.

4.The software should detect when a player reaches the finish line and declare them the winner.

Settings

1.The software should allow the user to change the music file of the game.

2.The software should allow the user to select the language and change other game options.

Performance Requirement

1.1.1 Response Time

The game should have a fast response time to ensure that players can access the options quickly. This includes the time taken to load pages, process user choices, and provide feedback to the user.

1.1.2 Concurrent Users:

The game should be able to handle multiple players concurrently. This includes the ability to support multiple modules without compromising the performance of the game

1.1.3 Scalability

The game should be scalable to handle an increasing number of players and game details over time. This includes the ability to add new features and functionalities without affecting the performance of the game.

1.1.4 Availability

The game should be available 24/7 without any downtime. This includes the ability to handle unexpected system failures and recover from them quickly.

1.1.5 Usability

The game should be easy to use and play. This includes clear and concise user interfaces and the ability to customize user preferences.

Design Constraints

1.1.5 Performance

The app must be designed to operate efficiently and quickly, with minimal response times for user requests and interactions.

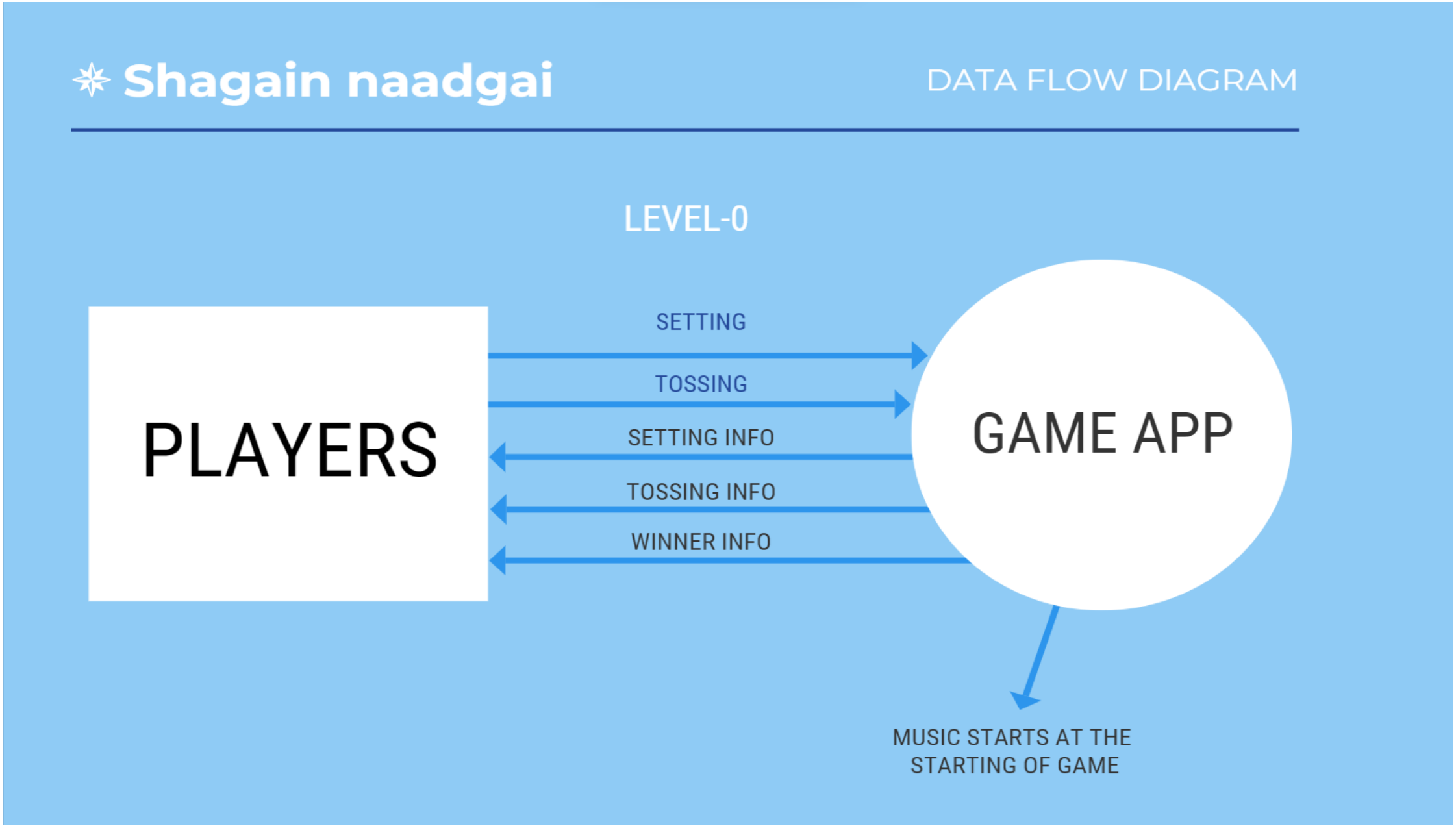
Scalability

The app must be designed to handle a large volume of users and data, with the ability to scale up or down as needed to accommodate changing usage patterns.

Usability

The app must be user-friendly and easy to access, with clear instructions and prompts for users.

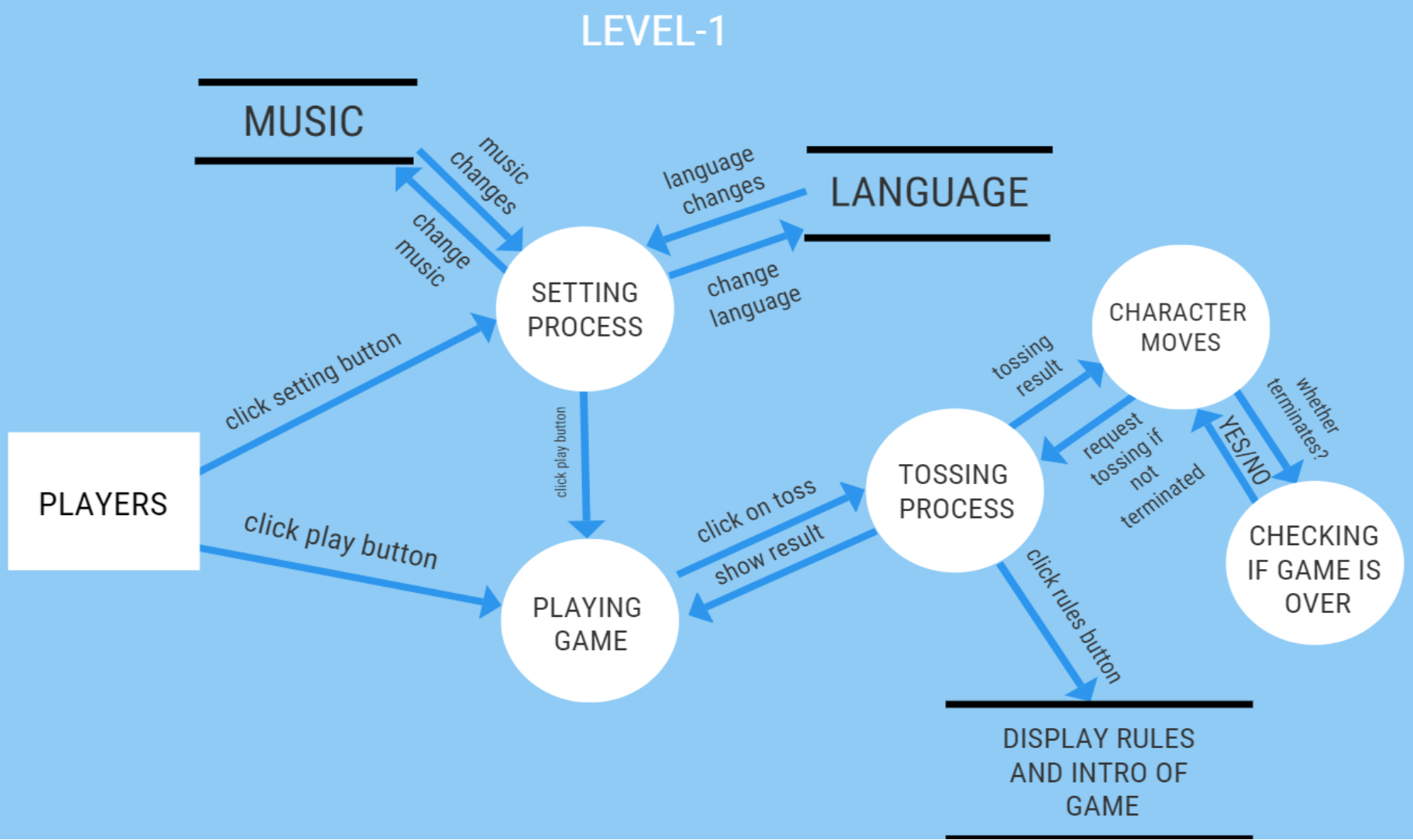
Data Flow Diagram



Data Flow Diagram

Shagain naadgai

DATA FLOW DIAGRAM



Estimations

Function Points

FACTOR	VALUE
BACKUP AND RECOVERY	2
DATA COMMUNICATION	0
DISTRIBUTED PROCESSING FUNCTIONS	0
CRITICAL PERFORMANCE	3
PERFORMANCE IN EXISTING AND HEAVILY UTILIZED ENVIRONMENT	0
ONLINE DATA ENTRY	0
INPUT TRANSACTION BUILT OVER MULTIPLE SCREENS	3
ILFS UPDATED ONLINE	0
INPUTS, OUTPUTS, FILES COMPLEX	0
INTERNAL PROCESSING COMPLEX	1
REUSABLE CODE	3
CONVERSIONS AND INSTALLATIONS INCLUDED	0
MULTIPLE INSTALLATIONS	2
CHANGE AND EASE OF USE	2

TOTAL=16

PARAMETERS	COUNT	SIMPLE	AVERAGE	DIFFICULT	TOTAL
EXTERNAL INPUT	5	5X3	0X4	0X6	15
EXTERNAL OUTPUT	4	4X4	0X5	0X7	16
EXTERNAL ENQUIRY	3	3X3	0X4	0X6	9
EXTERNAL INTERFACE FILES	0	0X5	0X7	0X10	0
INTERNAL LOGICAL FILES	3	0X7	3X10	0X15	30

COUNT TOTAL = 70

FP=COUNT TOTAL *[0.65+0.01*(TOTAL)]

=70*[0.65+0.01*16]

=56.7 FP

Efforts

Object type	Simple	Average	Difficult	Total
Screens	2X1	1X2	1X3	7
Reports	0X2	1X5	0X8	5
3GL Components	-	-	5X10	50

OP=62

%REUSE=40%

PRODUCTIVITY=13(NOMINAL)

NOP=OP*[100-%REUSE/100]

= 62[100-40/100]

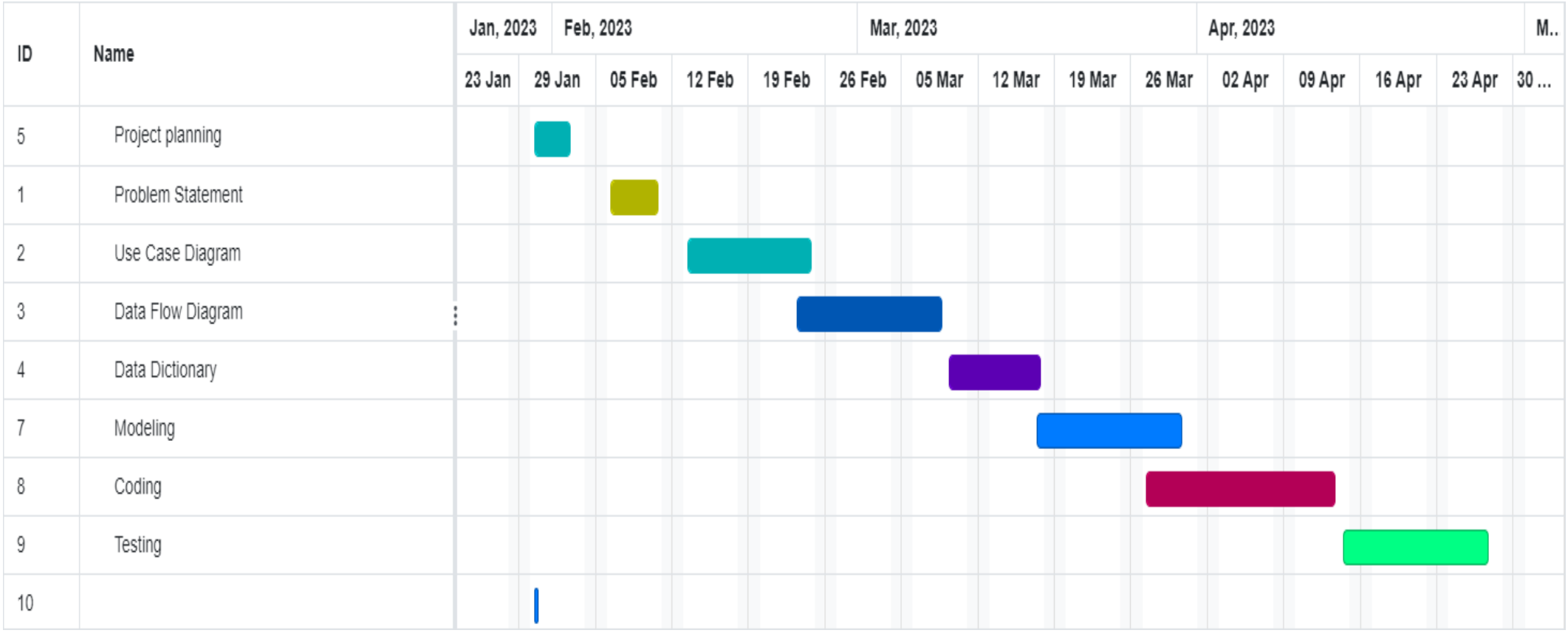
=37.2PM

EFFORT=NOP/PROD

=37.2/13

==2.86(APPROX) PM

Scheduling


ID
Name
Jan, 2023
Feb, 2023
Mar, 2023
Apr, 2023
M..
23 Jan
29 Jan
05 Feb
12 Feb
19 Feb
26 Feb
05 Mar
12 Mar
19 Mar
26 Mar
02 Apr
09 Apr
16 Apr
23 Apr
30 ...
5
Project planning
1
Problem Statement
2
Use Case Diagram
3
Data Flow Diagram
4
Data Dictionary
7
Modeling
8
Coding
9
Testing
10

Risk Management

RISKS	CATEGORY	PROBABILITY	IMPACT
TECHNICAL GLITCHES	TD	60	3
INADEQUATE TESTING	PS	40	3
NEW COMPETENT GAME	BU	50	2
SRS NOT DEFINED PROPERLY	PD	40	2
LESS REUSE THAN PLANNED	PS	40	3
IMPROPER WORKING CONDITON	DE	40	3
LACK OF MUTUAL UNDERSTANDING	ST	40	4
STAKEHOLDER REFUSES TO INVEST	SC	30	1
LACK OF TRAINING	DE	30	2
S/W QUALITY NOT UPTO MARK	PS	30	2
TECHNOLOGY NOT MEETING EXPECTATION	TD	20	1
NATURAL DISASTERS	SC	20	2
RELOCATION OF WORK PLACE	DE	10	3

Design

System Design

the game will be designed for windows computer systems. and will be divided into these several parts.

Game Logic

This module would handle the overall game logic, including determining when the game is over and calculating the final score.

User Interface

This module would provide the graphical user interface for the game, allowing users to interact with the game and receive feedback on the fortunes that are generated

Coin Toss

This module would handle the process of randomly tossing the four coins and determining the outcome based on the combination of the coins.

Fortune Table

This module would define the different fortune outcomes based on the combinations of the coins. It would need to be programmed with the various rules and conditions that dictate the fortunes for each combination.

data Design

the game won’t need a database, and the data model including the following entities.

Coin

This entity will store a single coin in the game, and would include attributes such as the type of animal it represents (horse, camel, sheep, or goat) and the current face that is showing.

Board

This entity will store the Board where the coin falls, this a Mongolian traditional design.

Fortune

This entity will store a fortune outcome in the game, and would include attributes such as the combination of coins that result in this fortune, the description of the fortune, and any associated point values.

Language This entity will store a language option in the game, and would include attributes such as the name of the language and the text strings for all the game elements that need to be displayed in that language.

Coding

1.pseudocode:Fortune box

1.Declare text as a Text component

2.Declare tossed as an integer

3.Declare description as a dictionary with integer keys and string values

4.Set text to the Text component of the current game object

Set tossed to 0

5**.While** true:

6. Set tossed to the value of the "tossed" variable in the "ShagaiNumberTextScript" script

7.Set the text of the "text" component to be the value of description[tossed]

8.END PROGRAM

pseudo code:shagai check zone

1. Declare side1_temee_count as an integer and initialize to 0

2. Declare side2_mori_count as an integer and initialize to 0

3. Declare side3_yamaa_count as an integer and initialize to 0

4. Declare side4_honi_count as an integer and initialize to 0

5. Declare side56_onkh_count as an integer and initialize to 0

6. Declare shagaiList as a List of GameObjects

7. Define Start method

8. Find all shagai objects with "shagai_tossing" tag and add them to shagaiList

9. Define OnTriggerStay method with parameter "other" of type Collider

10. **If** the tag of "other" is "side_shagai"

11. Find the parent shagai object of "other" collider

12. **If** shagaiList contains parentShagai and shagaiVelocity is zero and has not been counted yet

13. -Check which side of the shagai has collided and increment the corresponding count variable

14. **switch** by name of other gameobject

a.case 1:the side is "side1_temee":

increment the count of temee;

b.case 2:the side is "side2_mori of mori":

increment the count of mori

c.:case 3:the side is "side3_yamaa":

increment the count of yamaa;

d.case 4::the side is "side4_honi":

increment the count of honi; :

e.default: side5_onkh

increment the count of onkh

15. Set the parent shagai object's hasBeenCounted flag to true

16. **while** true:

17. **If** space key is pressed

a. Reset all count variables and text displays

b.Count the number of shagai that have stopped rolling and been counted

c. **If** all shagai have been counted

i. Update the text displays with the side counts

18.END PROGRAM

pseudo code:shagaiNumbertext

1. Define the class "ShagaiNumberTextScript"

2. Declare a "Text" variable "text" to hold the reference to the text component of the UI element

3. Declare static integer variables "tossed", "moriNumber", "temeeNumber", "honiNumber", "yamaaNumber", and "onkhNumber" to hold the values of the numbers and the side of the shagai

4. Declare string arrays "mnSides" and "enSides" to hold the Mongolian and English names of the sides of the shagai respectively

5. Declare a string variable "mode" to hold the current language mode

6. In the Start() function:

   a. Assign the "Text" component of the current game object to the "text" variable

   b. Initialize "tossed" variable to 0

   c. Get the current language mode from "LanguageButtonScript.mode" and assign it to "mode" variable

7. **while** true:

   a. **If** the "mode" variable is equal to "MN":

      i. Set the text of the "text" variable to the Mongolian names of the sides and their corresponding values

      ii. **If** the "onkhNumber" is greater than 0, set the "tossed" variable to -1 (which indicates that the shagai landed on "onkh" side)

      iii. Otherwise, calculate the value of the shagai using the formula provided and assign it to the "tossed" variable

   b. **Else:**

i. Set the text of the "text" variable to the English names of the sides and their corresponding values

ii. **If** the "onkhNumber" is greater than 0, set the "tossed" variable to -1

iii. Otherwise, calculate the value of the shagai using the formula provided and assign it to the "tossed" variable

8.end of codes inside while

9. END PROGRAM

pseudo code:shagai

1. Define public float variables for forceThreshold, maxVolume and minVolume, and an AudioSource variable called audioSource.

2. Define public Vector3 variables for shagaiVelocity and initialPosition, and a public bool variable called hasBeenCounted.

3. In Start() function, set audioSource to the AudioSource component of the game object.

4. Set rb to the Rigidbody component of the game object, and hasBeenCounted to false.

5. Generate random values for dirX, dirY and dirZ between 0 and 500, and add torque to the rigid body using these values.

6. Set initial Position to the current position of the game object.

7. In OnCollisionEnter() function, calculate the volume of the collision impulse using Mathf.Clamp01() function, and set the volume of audioSource using Mathf.Lerp() function.

8. Play the audio Source.

9. **while** true:

10. update shagaiVelocity variable to the current velocity of the rigid body.

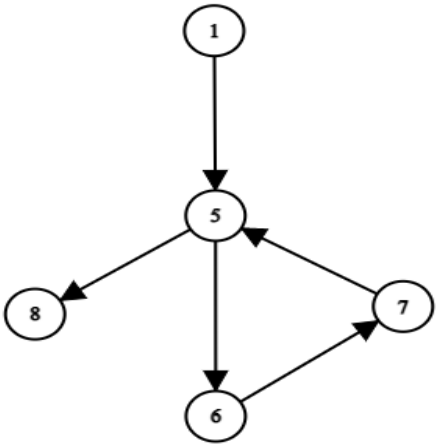
11. Check **if** the Space key has been pressed, and if so, reset the position of the game object to initial Position, generate random values for dirX, dirY, dirZ, rotX, rotY and rotZ, set the rotation of the game object using Quaternion.Euler() function, add force to the rigid body using transform.up and a value of 500, add torque to the rigidbody using dirX, dirY and dirZ, and set hasBeenCounted to false.

12. END PROGRAM

Testing

I. Fortune box

Graph:



R=number of regions=2

Cyclometric complexity=2

Test Case:

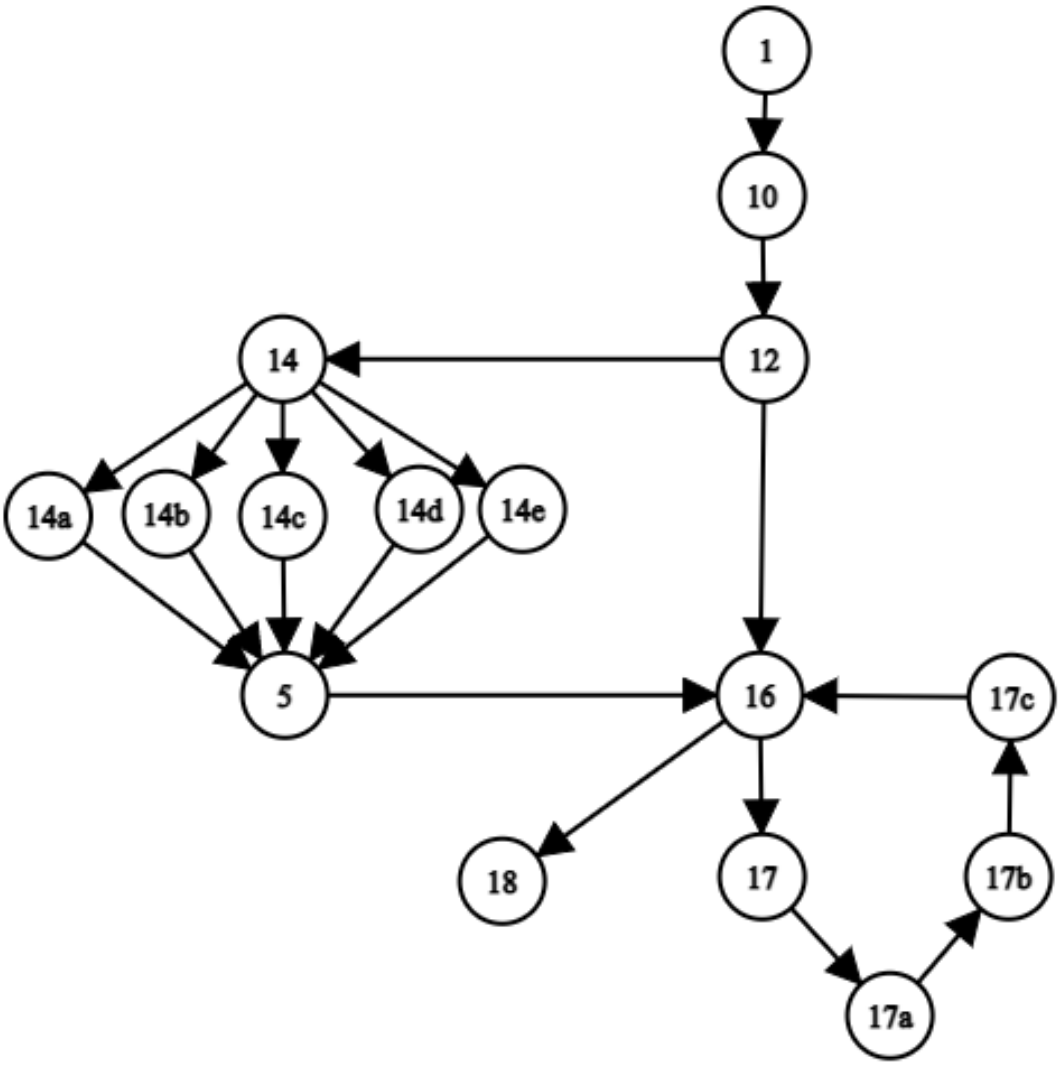
Project Name	TMT
Module Name	Fortune Box
Created Date	10/04/2023

Test Case ID	Scenario Description	Pre-Condition	Steps to Execute	Expected Result	Actual Result	Status
1	Fortune when shagais are not stopped	The game is loaded and shagais	Toss by pressing space	Shows message “please toss”	Shows message “please toss”	PASSED

		are not stiopped				
2	Fortune when shagais are stopped	Shaigais are stopped (velocity 0)	1)Toss by pressing space 2) Wait until all shagais stop	shows fortune according to number of upper faces of shagais	shows fortune according to number of upper faces of shagais	PASSED

ii.Shagai Check Zone

Graph:



R=number of regions=7

Cyclometric complexity=7

Paths: P1 =1->10->12->14->14a->15->16->17->17a->17b->17c->16->18

P2=> 1->10->12->14->14b->15->16->17->17a->17b->17c->16->18

P3=>1->10->12->14->14c->15->16->17->17a->17b->17c->16->18

P4=>1->10->12->14->14d->15->16->17->17a->17b->17c->16->18

P5=>1->10->12->14->14e->15->16->17->17a->17b->17c->16->18

P6=>1->10->12->14->14b->15->16->16->18

P7=>1->10->12->16->17->17a->17b->17C->16->18

P8=>1->10->12->16->16->18

P9 =1->10->12->14->14a->15->16->17->17a->17b->17c->16->18

P10=>1->10->12->14->14b->15->16->18

P11=>1->10->12->14->14c->15->16->18

P12=>1->10->12->14->14d->15->16->18

P13=>1->10->12->14->14e->15->16->18

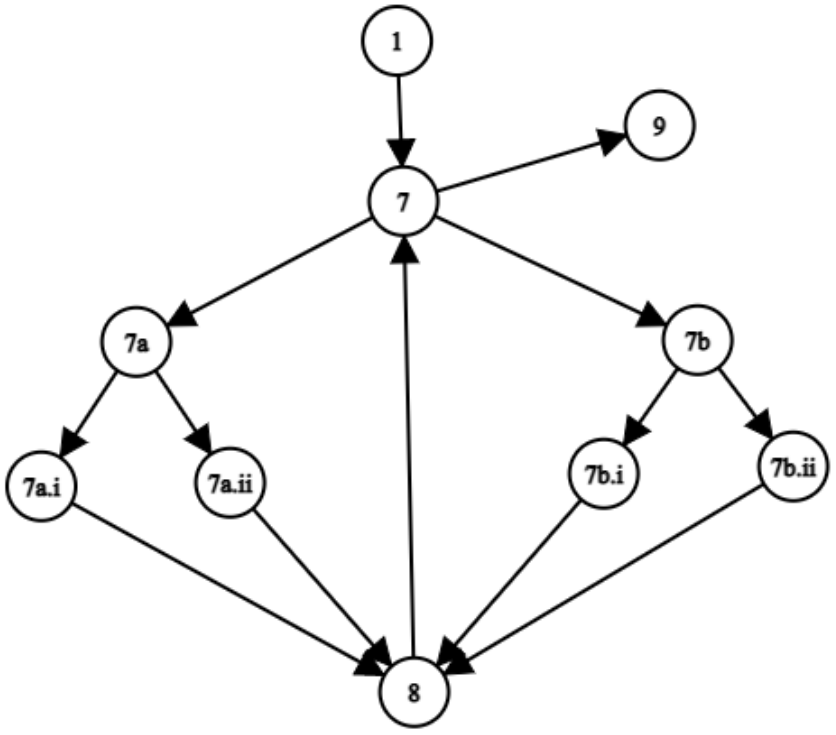
Test Case:

Project Name	TMT
Module Name	Shagai Check Zone
Created Date	16/04/2023

Test Case ID	Scenario Description	Pre-Condition	Steps to Execute	Expected Result	Actual Result	Status
1	Collition with shagais	The game is loaded	Toss by pressing space	When shagais reach board or zone, shagais collide and bounce	When shagais reach board or zone, shagais collide and bounce	PASSED
2	Detection of Shagais faces	Shaigais are stopped (velocity 0)	1)Toss by pressing space 2) Wait until all shagais stop	Counts exact number of faces of shagais then saves in public	Counts exact number of faces of shagais then saves in public	PASSED

iii.shagaiNumbertext

Graph:



Cyclometric complexity=5

Paths: P1 => 1->7->7a->7a.i->8->7->9

P2 => 1->7->7b->7a.ii->8->7->9

P3 => 1->7->7b->7b.i->8->7->9

P4=> 1->7->7b->7b.ii->8->7->9

P5 => 1->7->9

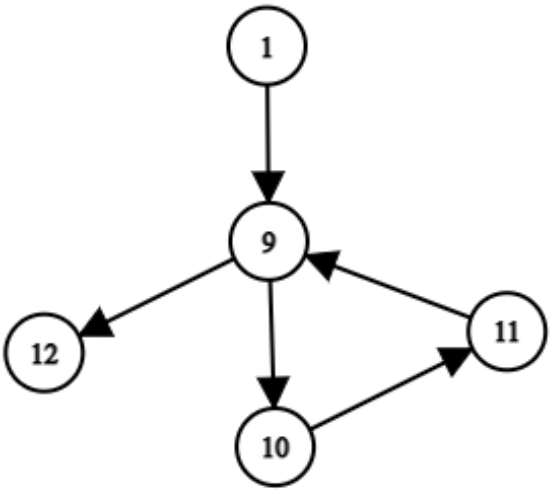
Test Case:

Project Name	TMT
Module Name	Shagai Number Text
Created Date	13/04/2023

Test Case ID	Scenario Description	Pre-Condition	Steps to Execute	Expected Result	Actual Result	Status
1	Counts of faces of shagais when they are moving	The game is loaded and shagais are not stopped	Toss by pressing space	Shows 0 for each number of faces	Shows message “please toss”	PASSED
2	Fortune when shagais are stopped	Shaigais are stopped (velocity 0)	1)Toss by pressing space 2) Wait until all shagais stop	shows numbers of faces of shagais	shows numbers of faces of shagais	PASSED

iv.shagai

Graph



R=number of regions=2

Cyclometric complexity=2

Paths:

P1 => 1->9->10->11->9….

P2 => 1->9->12

Test Case:

Project Name	TMT
Module Name	Shagai
Created Date	16/04/2023

Test Case ID	Scenario Description	Pre-Condition	Steps to Execute	Expected Result	Actual Result	Status
1	Voice of Collision	The game is loaded and shagais are not stopped	Toss by pressing space	Shagais makes sound according to the force of collision where and when they make collision with other objects	Shagais makes sound according to the force of collision where and when they make collision with other objects	PASSED
2	Start tossing when space is pressed	The game is loaded and shagais are tossed	Toss by pressing space	Shagais position will change to its initial position, rotates randomly, adds force downwards	Shagais position will change to its initial position, rotates randomly, adds force downwards	PASSED
3	Gravity	The game is loaded and shagais are tossed	Toss by pressing space	Gravity gives force to shagais and bounces on the board like real life	Gravity gives force to shagais and bounces on the board like real life	PASSED

References

1. UNITY platform
    a. https://unity.com/
    b. https://forum.unity.com/
    c. https://learn.unity.com/
2. Youtube
    a. https://www.youtube.com/
    b. https://www.youtube.com/watch?v=D9qt1cTH2r0

ame will be designed for windows computer systems. and , will be divided into these several parts.

Game Logic

This module would handle the overall game logic, including determining when the game is over and calculating the final score.

User Interface

This module would provide the graphical user interface for the game, allowing users to interact with the game and receive feedback on the fortunes that are generated

Coin Toss

This module would handle the process of randomly tossing the four coins and determining the outcome based on the combination of the coins.

Fortune Table

This module would define the different fortune outcomes based on the combinations of the coins. It would need to be programmed with the various rules and conditions that dictate the fortunes for each combination.

Language

This module would handle the different language options that you mentioned. It would allow users to switch between English and Mongolian languages.

data Design

the game won't need a database, and the data model including the following entities.

Coin

This entity will store a single coin in the game, and would include attributes such as the type of animal it represents (horse, camel, sheep, or goat) and the current face that is showing.

Board

This entity will store the Board where the coin falls, this a Mongolian traditional design.

Fortune

This entity will store a fortune outcome in the game, and would include attributes such as the combination of coins that result in this fortune, the description of the fortune, and any associated point values.

Language

This entity will store a language option in the game, and would include attributes such as the name of the language and the text strings for all the game elements that need to be displayed in that language.

Coding

1.pseudocode:Fortune box

pseudo code:shagai

```
IMPORT Unity Engine
IMPORT System.Collections
IMPORT System.Collections.Generic

CLASS ShagaiScript
    VARIABLE forceThreshold : Float = 5.0
    VARIABLE maxVolume : Float = 1.0
    VARIABLE minVolume : Float = 0.1
    VARIABLE audioSource : AudioSource

    VARIABLE rb : Rigidbody
    VARIABLE shagaiVelocity : Vector3
```

```
    VARIABLE hasBeenCounted : Boolean
    VARIABLE initialPosition : Vector3

    FUNCTION Start()
        audioSource = GET_COMPONENT(AudioSource)
        rb = GET_COMPONENT(Rigidbody)
        hasBeenCounted = false

        VARIABLE dirX : Float = Random.Range(0, 500)
        VARIABLE dirY : Float = Random.Range(0, 500)
        VARIABLE dirZ : Float = Random.Range(0, 500)

        rb.AddTorque(dirX, dirY, dirZ)
        initialPosition = GET_POSITION(transform)

    FUNCTION OnCollisionEnter(collision : Collision)
        VARIABLE volume : Float =
Mathf.Clamp01(GET_MAGNITUDE(collision.impulse) / forceThreshold)
        audioSource.volume = Mathf.Lerp(minVolume, maxVolume,
volume)
        audioSource.Play()

    FUNCTION Update()
        shagaiVelocity = GET_VELOCITY(rb)

        IF GET_KEY_DOWN(KeyCode.Space)
            VARIABLE dirX : Float = Random.Range(0, 500)
```

```
                VARIABLE dirY : Float = Random.Range(0, 500)
                VARIABLE dirZ : Float = Random.Range(0, 500)

                SET_POSITION(transform, initialPosition)

                VARIABLE rotX : Float = Random.Range(0, 360)
                VARIABLE rotY : Float = Random.Range(0, 360)
                VARIABLE rotZ : Float = Random.Range(0, 360)

                SET_ROTATION(transform, Quaternion.Euler(rotX,
rotY, rotZ))

                rb.AddForce(GET_UP(transform) * 500)
                rb.AddTorque(dirX, dirY, dirZ)

                hasBeenCounted = false
            END IF
        END FUNCTION
END CL
```