INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Percussive Arts Society

Master Document
Construcción de software y toma de decisiones TC2005B, Grupo 400 Octavio
Navarro Hinojosa Dr. Esteban Castillo Juarez Gilberto Echeverría Furió

Manuel Barrera López Portillo A01570669
Juan Muniani Otero A01781341
Miguel Angel Bustamante Perez A01781583

Abstract:

This document presents the statements and organization of the present game development during the past months and summarizes the design of the game and its connections for data gathering.

General Design:

We design our game under the following design philosophy:

- The game is a didactic tool that records highscores, players username, and other certain data to interpret certain characteristics.
- A database to store and recover certain data that will be pertinent to the functioning of a website and a videogame.
- A website made to inform structure and display information relevant to the game and to the subject in hand that is: Health issues regarding percussionists and people interested in percussive instruments.

SCRUM ONE PAGE
The following section describes the functional requirements of the project.

- Add player controller script.
  - The game file has functioning controls that permit the player to interact with the game as intended. This means that certain keyboard keys interact with the specified triggers adding certain scores.
- Add track manager script.
  - The track manager controls the order that certain music notes are played at a certain time. In this track the project must work with the player controller script.
- Create midi samples.

  The midi files are necessary to define the tracks of music used in the files needed to play the game.
- Add columns script.
  - The script needed to define the space used for the sprites rolling through the game.
- Add scoring script
  - Program that determines the amount of points being scored through the midi files length.
- Add combo / score hud
  - The combo multiplier works by making every scoring action larger by multiplying by a certain factor. This makes the player score more points for the more correct notes inserted in a row.

- Design the main menu.
  - The main menu is a script that allows the user to enter the game and select a level.
- Create website:
  - The website includes functioning javaScripts, API, CSS and HTML files to permit the proper usage and communication of the Video game and the database.
- Setup MySQL queries.
  - The script that permits the communication of the website and video game with the database.
- Setup MySQL tables.
  - create the scheme that permits database operations
- Setup API.
  - Creating the relevant request methods that help communicate in the backend the website

Non functional requirements

- Creating the sprites relevant to the game such as Jazzcat and the catto battery
- Creating visual assets that add to the game experience. Such as buttons, web assets, etc.
- Creación de documento CSS y HTML para el sitio web.

During the period of game development, there were 5 sprints in which the following activities where realized

1. Week 1 Sprint. Man hours of work: 10 hs over the week. Scrum Master Juan Muniani
   a. Create level Backgrounds 5 hrs
   b. Create Audience Sprite 2 hrs
   c. Create miss, good, great sprites 1hr
   d. Create drum set sprites. 1 hr
2. Week 2 Sprint. Man hours of work: 10 hrs over the week Scrum Master Miguel Bustamante
   a. Create columns scripts 2 hrs
   b. Create score scripts 2 hrs
   c. Database Scheme 2 hrs
3. Week 3 Sprint. Man hours of work 10 hrs over the week Scrum Master Manuel Barrera
   a. Create Fish Sprite 2 hrs
   b. Create menu and end level hud 2 hrs
   c. Create visual assets 2 hrs
   d. Create combo script 4 hrs
4. Week 4 Sprint. Man hours of work over the week 40hrs over the week Scrum Master Juan Muniani
   a. Create Midi Samples 10hrs

    b. Create Final database scheme 12 hrs
    c. Create Unit Api 10hrs
    d. Create website api 8hrs
5. Week 5 Sprint. Man hours: 23 hrs over the week Scrum Master: Miguel Bustamante
    a. Polishing of the different aspects of the website 8hrs
    b. Perfecting the website. 8hrs
    c. Making the interconnections between the website, the database and the video game. 7hrs

In total we worked around 96 hours. As the past statistics show.

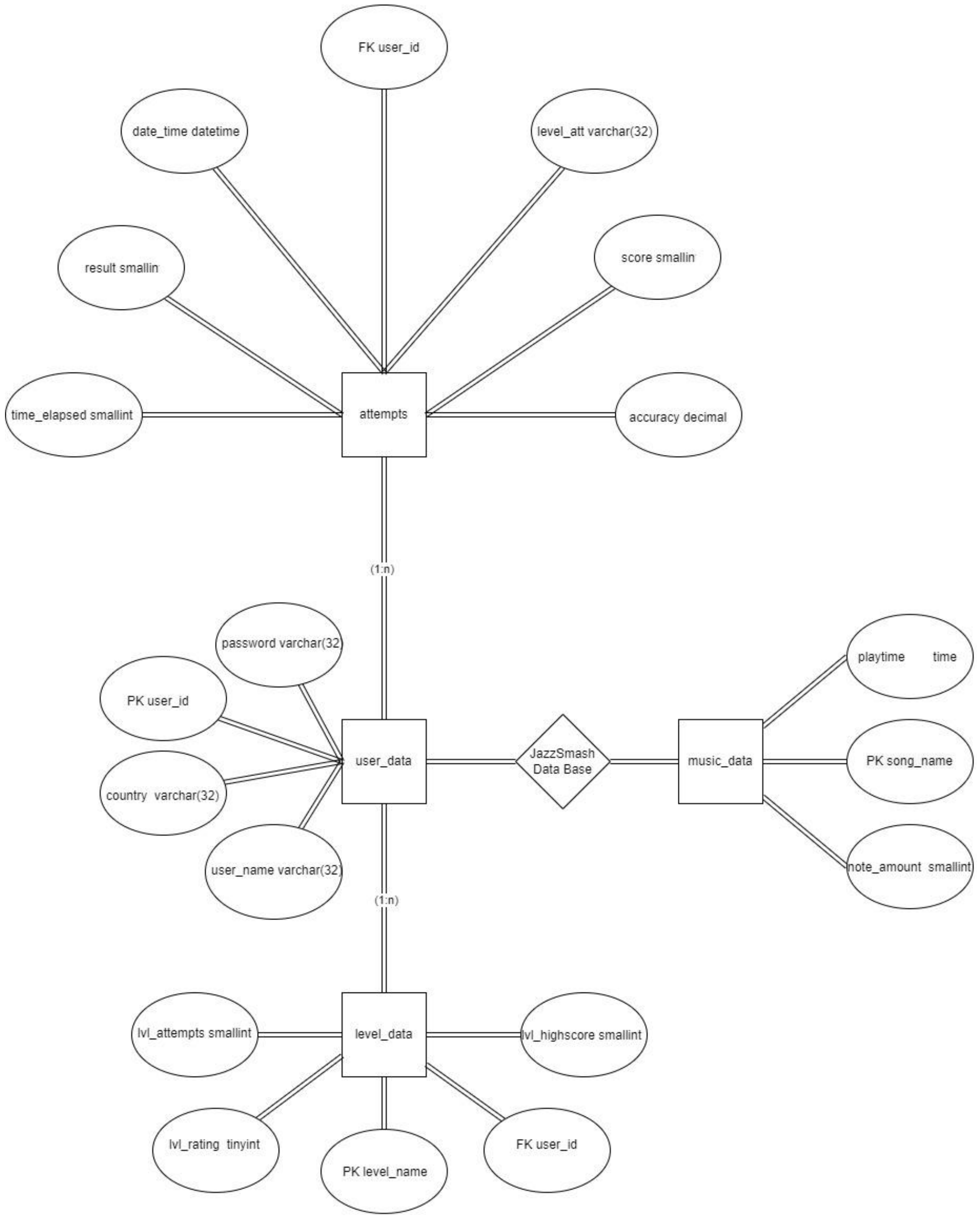
Persons	Time developing
Juan Muniani	45 hrs
Miguerl Bustamante	40 hrs
Manuel Barrera	11 hrs

Datatable Justification:

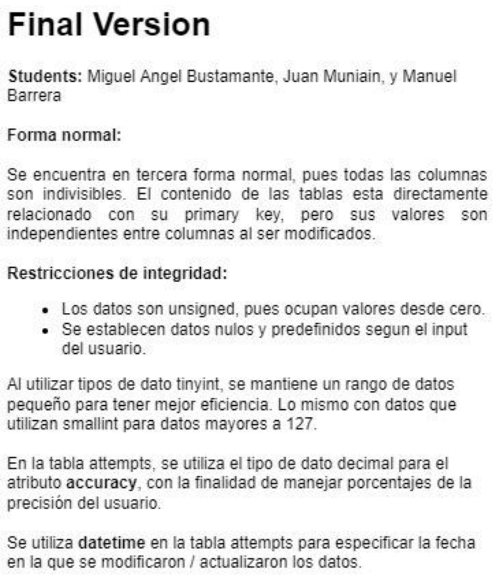
The database table is in the third normal form (3NF) according to the following statements:

- Each table of the scheme has its own primary key.
- Each table has no complex keys, each primary key is one valued.
- Every value established and defined is atomic. The only one that in our naming that could be seen as not atomic is music_name as it contains the title and author of the music, but in this case is the name of the file available.
- Our database script has anti redundancy measures that prohibit the same username from being created.
- There are no transitive tables in our database.

Entity Relationship Diagrams



Procedure Design



Final Version

Students: Miguel Angel Bustamante, Juan Muniain, y Manuel Barrera

Forma normal:

Se encuentra en tercera forma normal, pues todas las columnas son indivisibles. El contenido de las tablas esta directamente relacionado con su primary key, pero sus valores son independientes entre columnas al ser modificados.

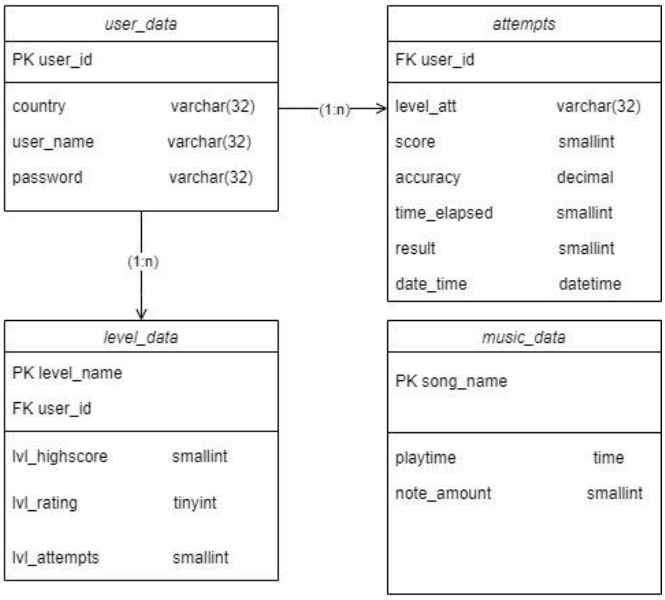
Restricciones de integridad:

- Los datos son unsigned, pues ocupan valores desde cero.
- Se establecen datos nulos y predefinidos segun el input del usuario.

Al utilizar tipos de dato tinyint, se mantiene un rango de datos pequeño para tener mejor eficiencia. Lo mismo con datos que utilizan smallint para datos mayores a 127.

En la tabla attempts, se utiliza el tipo de dato decimal para el atributo **accuracy**, con la finalidad de manejar porcentajes de la precisión del usuario.

Se utiliza **datetime** en la tabla attempts para especificar la fecha en la que se modificaron / actualizaron los datos.



user_data	
PK user_id	
country	varchar(32)
user_name	varchar(32)
password	varchar(32)

attempts	
FK user_id	
level_att	varchar(32)
score	smallint
accuracy	decimal
time_elapsed	smallint
result	smallint
date_time	datetime

level_data	
PK level_name	
FK user_id	
lvl_highscore	smallint
lvl_rating	tinyint
lvl_attempts	smallint

music_data	
PK song_name	
playtime	time
note_amount	smallint

User Stories

User story #1 Player controller script As a player I want to be able to input commands and visualize feedback from my attacks.	
Validation ● Input commands set in the script work accordingly. Points and sprites work based on user input.	Point value: 100 maximum 0 minimum

User story #2 Add track manager script As a player I want a smooth gameplay experience when interacting with enemy sprites following the rhythm of the song.	
Validation ● Enemy sprites move according to the beat of the song. ● Sprites spawn at their given time. ● Attacking enemies at the correct time effectively destroys them. ● Enemies will despawn after failing to attack them.	Point value: 100 maximum 0 minimum

<table>
<tr><td colspan="2">User story #3 Create midi samples
As a player
I want to have a musical system that interacts with the player and the sprites in order to have a fluid experience with the game.</td></tr>
<tr><td>Validation
● Timing for every drum kit part should create an accurate sequence of percussion.
● Sprites should spawn according to their corresponding midi sequence.</td><td>Point value: 100 maximum 0 minimum</td></tr>
</table>

<table>
<tr><td colspan="2">User story #4 Add columns script
As a player
I want to be able to differentiate and destroy enemies based on their corresponding lane spawn.</td></tr>
<tr><td>Validation
● Enemies are restricted to their corresponding lane
● Certain inputs effectively destroy their corresponding enemy.
● Enemies spawn as prefabs in a given order and time.</td><td>Point value: 100 maximum 0 minimum</td></tr>
</table>

<table>
<tr><td colspan="2">User story #5 Add scoring script
As a player
I want to be able to keep track of my performance throughout a level, and be able to obtain combos, and calculate accuracy.</td></tr>
<tr><td>Validation
● 'Good', 'perfect', and 'miss' sprites spawn accordingly.
● Score updates accordingly when performing attacks.
● Combo resets when missing attacks.
● Accuracy is calculated at the end of the level based on accumulated hit and miss data.</td><td>Point score: max 100 points min 0 points</td></tr>
</table>

<table>
<tr><td>User story #6 Add combo / score hud
As a player
I want to be able to visualize my results at the end of a level.</td></tr>
</table>

Validation ● Score text updates according to the attack result (good, perfect, miss). ● Information is displayed in an intuitive, colorful way for easy feedback.	Point score: max 100 points min 0 points

User story #7 Design Main Menu As a player I want a main menu that displays level options and difficulties.	
Validation ● The game should have a main menu which lets the player display a level select submenu. ● The game has to have another menu that allows the player to select the level difficulty.	Point score: max 100 points min 0 points

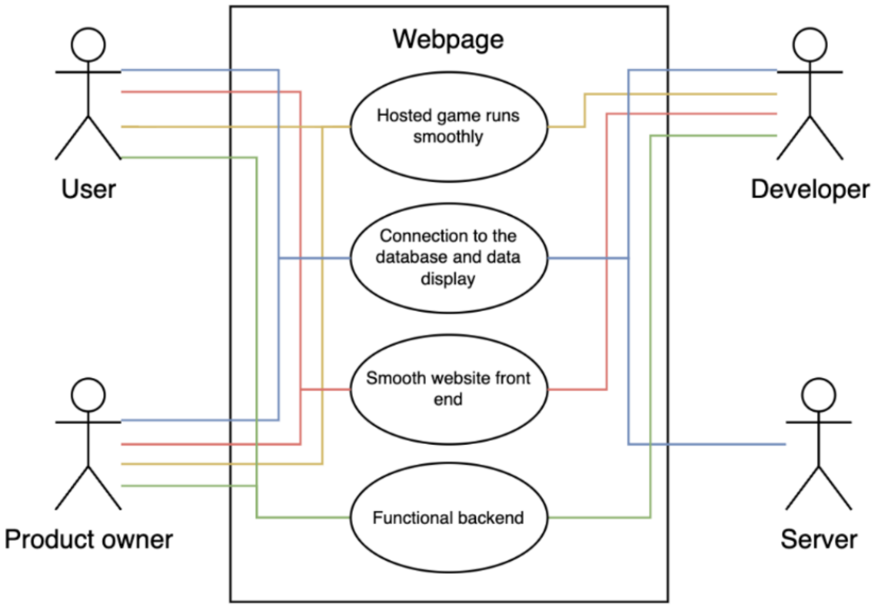
User story #8 Website Design As a user I want that each time I enter the web page, I can access medical information / recommendations about drumming.	
Validation ● The website loads correctly. ● The website has an appropriate and intuitive design. ● The game runs smoothly within the site.	Point score: max 100 points min 0 points

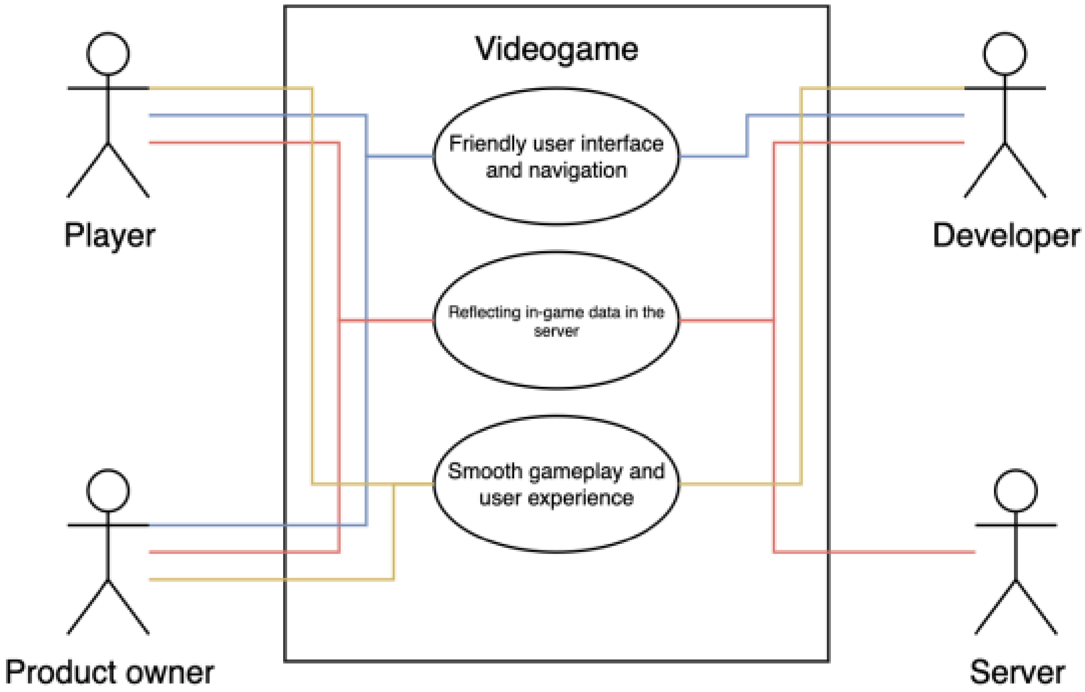
User story #9 Setup MySQL queries As a database admin I want to be able to insert, extract, update, and delete information from the database in an efficient way.	
Validation ● Integrity restrictions are followed (updated information is not invalid, primary and secondary keys are set to prevent data repetition) ● Data manipulation in one table affects related tables which share foreign keys.	Point score: max 100 points min 0 points

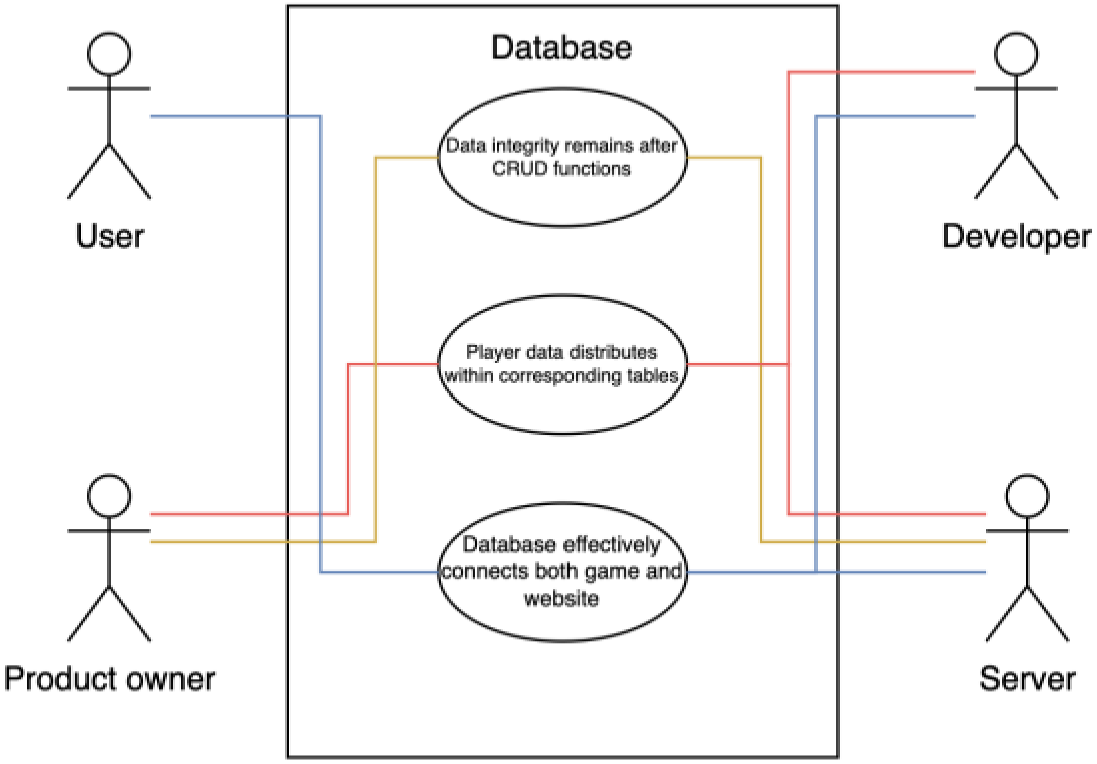
User story #10 Setup MySQL tables As a database admin I want to be able to distribute information within tables where each represents game information.	
Validation ● Data is structured / normalized based on normal forms. ● Data is distributed within its entity / table	Point score: max 100 points min 0 points

User story #11 Setup web-database link As a website admin I want to be able to extract information from the database and display it with statistics.	
Validation ● Player information generated after ending a level is reflected in the webpage. ● High score information is updated accordingly.	Point score: max 100 points min 0 points

UML Diagram



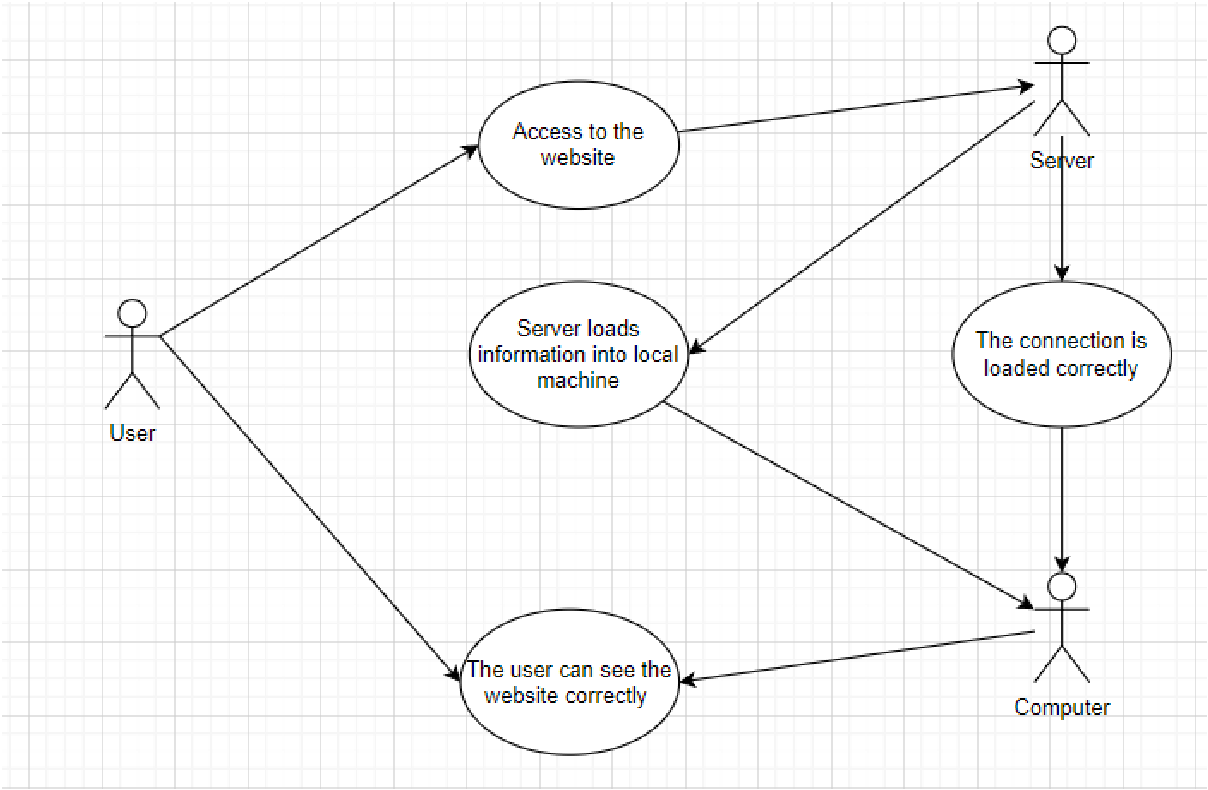

Videogame
Player
Developer
Friendly user interface and navigation
Reflecting in-game data in the server
Smooth gameplay and user experience
Product owner
Server


Database
User
Developer
Data integrity remains after CRUD functions
Player data distributes within corresponding tables
Database effectively connects both game and website
Product owner
Server

Use case Tables

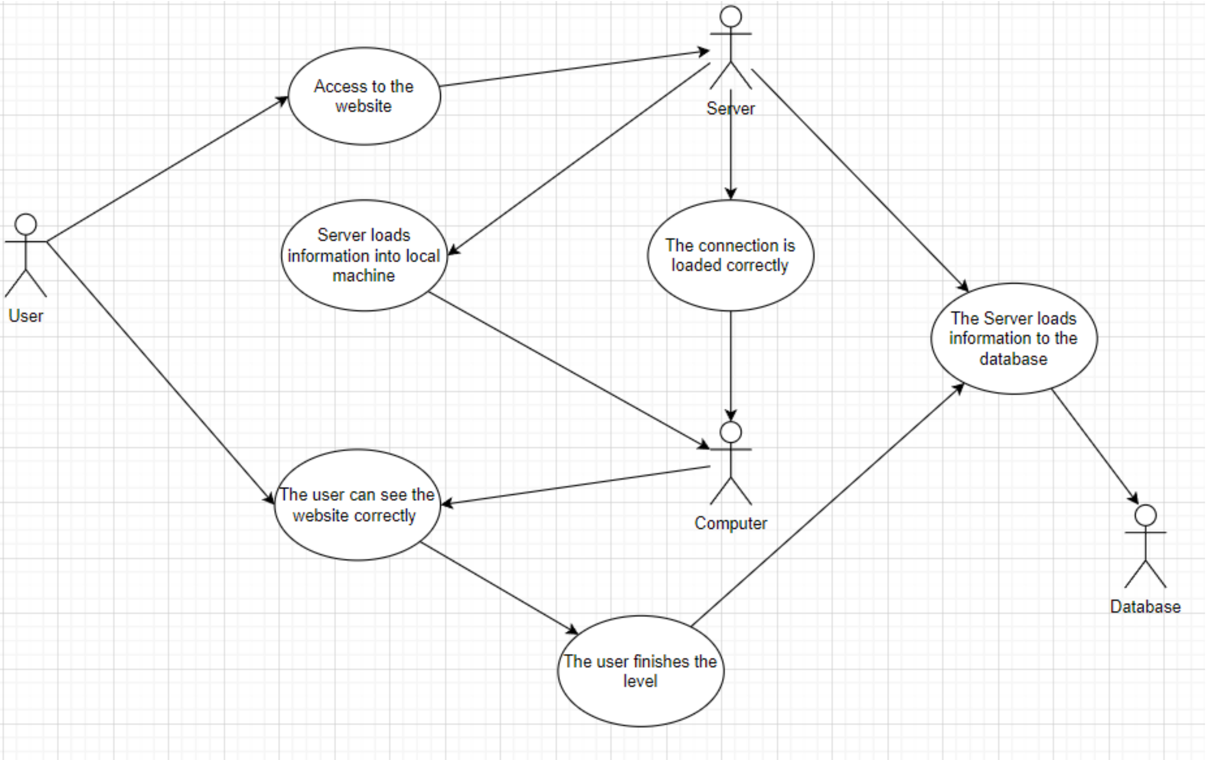
<table>
<tr><td>Use case name</td><td>Hosted game runs smoothly</td></tr>
<tr><td>Related Requirements</td><td>Requirements of Video Game development</td></tr>
<tr><td>Goal in Contex</td><td>To assure the full functionality of the Videogame in our web page</td></tr>
<tr><td>Preconditions</td><td>The Web Page must have its related assets fully included and as well functional</td></tr>
<tr><td>Successful end condition</td><td>The video game cycle runs correctly. The web page is interactive and has no visual glitches The relevant data related to the database is taken and recorded correctly.</td></tr>
<tr><td>Primary Actors</td><td>User</td></tr>
<tr><td>Secondary Actor</td><td>Server</td></tr>
<tr><td>Trigger</td><td>The user access the website and then can access the game and relevant information of the game</td></tr>
<tr><td></td><td>
<table>
<tr><td>Step</td><td>Execution</td></tr>
<tr><td>1</td><td>the User enters the web page</td></tr>
<tr><td>2</td><td>The server loads the information relevant to the webpage (html js css and game)</td></tr>
<tr><td>3</td><td>The page is loaded successfully</td></tr>
<tr><td>4</td><td>The user can interact with the page: move the page, click the relevant information without glitches.</td></tr>
<tr><td>5</td><td>The data base makes a successful</td></tr>
</table>
</td></tr>
</table>

	connection
6	The database updates itself, getting the new imputed data.
7	The website stays in this state without problems
8	The user can close the website


User
Access to the website
Server
Server loads information into local machine
The connection is loaded correctly
Computer
The user can see the website correctly

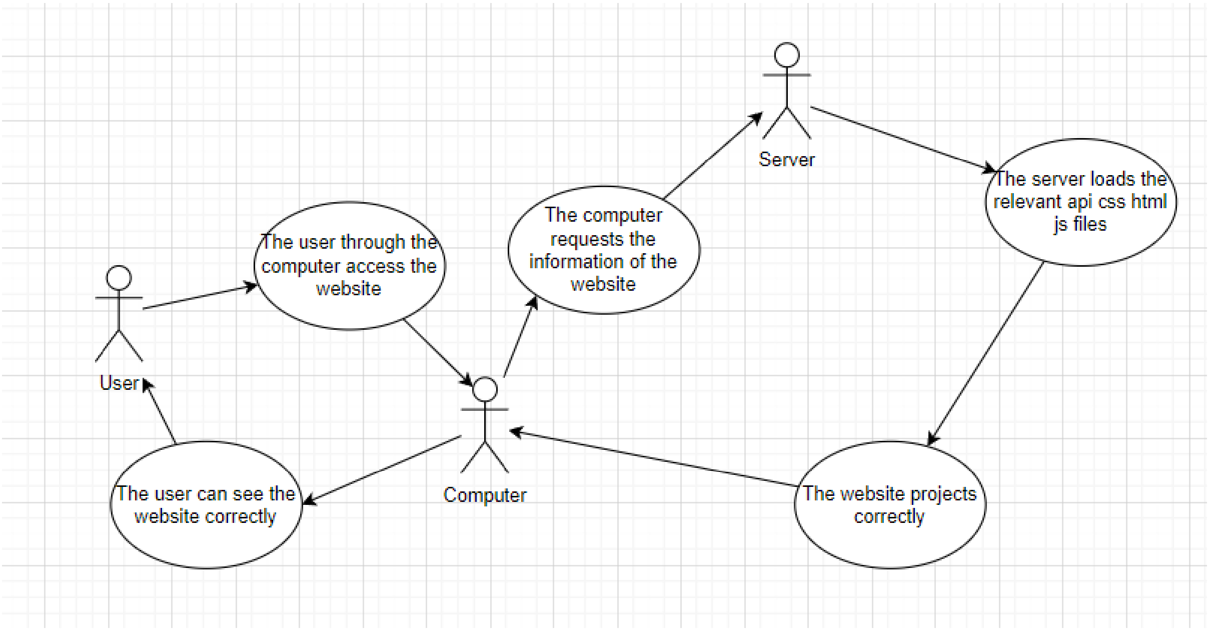
Use case name	Connection to the database and data display

<table>
<tr><td>Related Requirements</td><td>Requirements related to the creation of the database</td></tr>
<tr><td>Goal in Contex</td><td>To record all highscores of the game over time to understand how people interact with percussion skills like rhythm detection</td></tr>
<tr><td>Preconditions</td><td>The database must have the correct parameters for its correct functioning such as normalization.</td></tr>
<tr><td>Successful end condition</td><td>The database has a proper connection to the webpage and receives through the internet the Highscores of each player</td></tr>
<tr><td>Primary Actors</td><td>Server</td></tr>
<tr><td>Secondary Actor</td><td>Database developer</td></tr>
<tr><td>Trigger</td><td>The player finishes playing a level of the game the High Score is recorded</td></tr>
<tr><td></td><td>
<table>
<tr><td>Step</td><td>Execution</td></tr>
<tr><td>1</td><td>The user loads the web page and plays the game</td></tr>
<tr><td>2</td><td>The user finishes a level</td></tr>
<tr><td>3</td><td>The web page communicates with the database</td></tr>
<tr><td>4</td><td>The web page inserts the username date and highscore of the user</td></tr>
<tr><td>5</td><td>The insertion is confirmed as correct</td></tr>
</table>
</td></tr>
</table>


Access to the website
Server
Server loads information into local machine
The connection is loaded correctly
User
The Server loads information to the database
Computer
The user can see the website correctly
Database
The user finishes the level

<table>
<tr><td>Use case name</td><td>Smooth website front end</td></tr>
<tr><td>Related Requirements</td><td>The design of the Html and css files of the website</td></tr>
<tr><td>Goal in Contex</td><td>To permit a functional, well design and informative website for the user</td></tr>
<tr><td>Preconditions</td><td>The correct arrangement in the server of the corresponding html css and js scripts.</td></tr>
<tr><td>Successful end condition</td><td>The website should load with the correct design without visual glitches or design errors</td></tr>
<tr><td>Primary Actors</td><td>Server</td></tr>
<tr><td>Secondary Actor</td><td>Errors</td></tr>
<tr><td>Trigger</td><td>The user accesses the server</td></tr>
<tr><td></td><td><table><tr><td>Step</td><td>Execution</td></tr></table></td></tr>
</table>

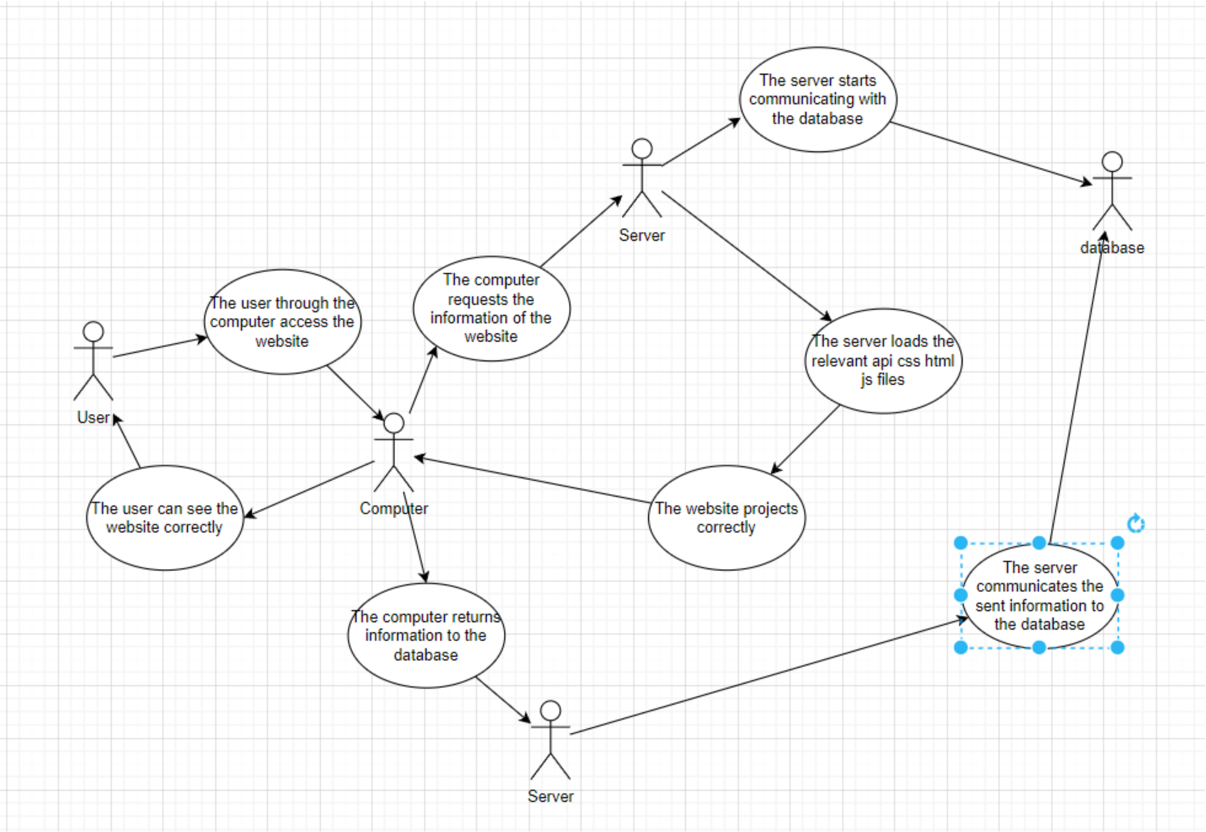
<table>
<tr>
<td></td>
<td>
<table>
<tr><td>1</td><td>The user accesses the website address</td></tr>
<tr><td>2</td><td>The html and css must load correctly</td></tr>
<tr><td>3</td><td>The website displays correctly</td></tr>
</table>
</td>
</tr>
</table>


Server
The server loads the relevant api css html js files
The computer requests the information of the website
The user through the computer access the website
User
Computer
The user can see the website correctly
The website projects correctly

Use case name	Functional Backend
Related Requirements	Smooth frontend functional
Goal in Contex	The augmentation of the functionality of the website to a more interactive version
Preconditions	The related js modules must be functional
Successful end condition	The interactive displays such as the game space must appear correctly
Primary Actors	Server

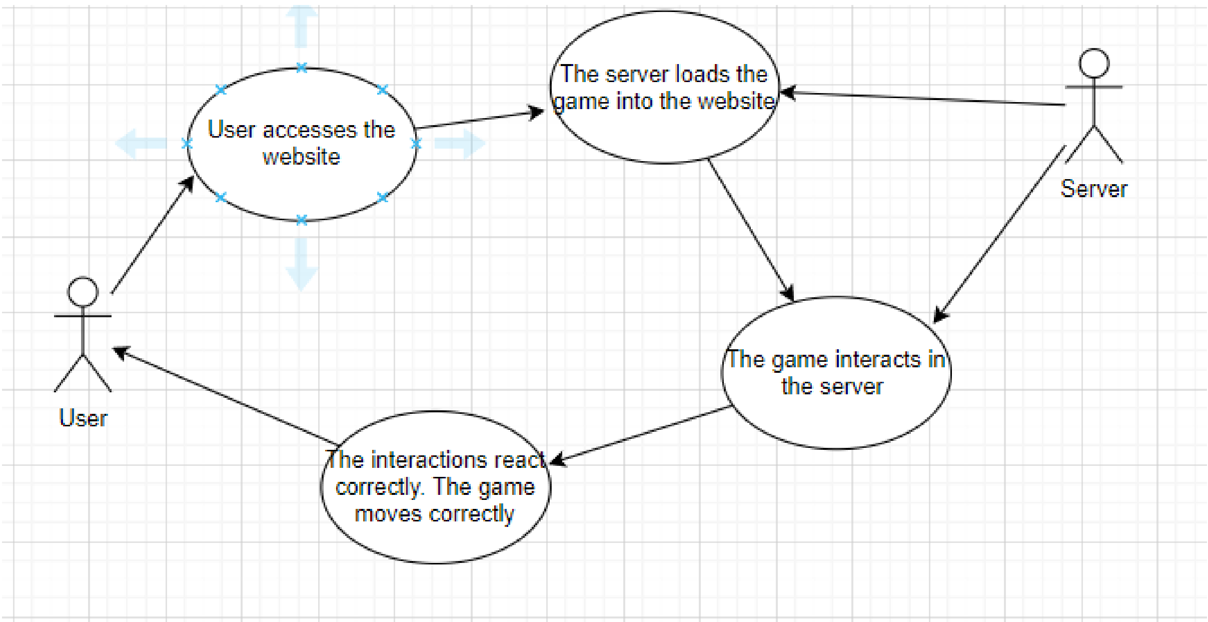
Secondary Actor	User Developer
Trigger	The user accesses the website

Step	Execution
1	the user access the website
2	the server loads into the computer the relevant api css html and js files
3	The server connects the to the database and computer to star communication
4	The server is connected.


The server starts communicating with the database
Server
database
The computer requests the information of the website
The user through the computer access the website
User
The server loads the relevant api css html js files
The user can see the website correctly
Computer
The website projects correctly
The server communicates the sent information to the database
The computer returns information to the database
Server

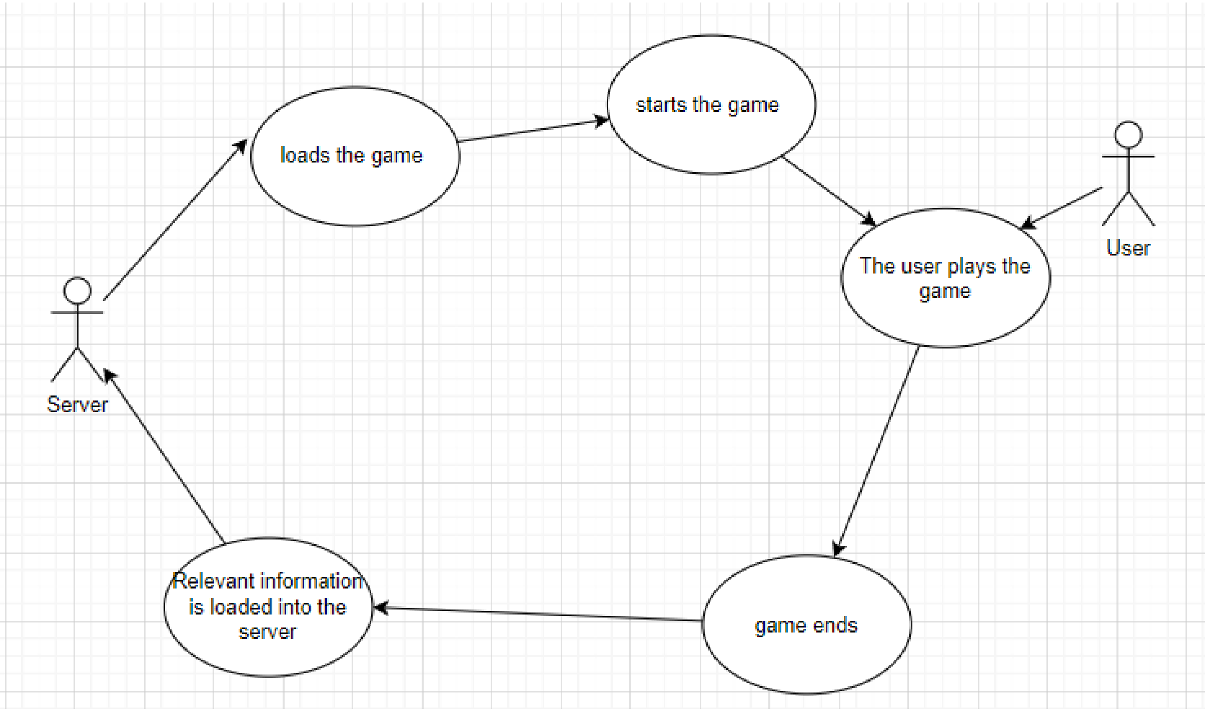
Use Case tables: Videogame

<table>
<tr><td>Use case name</td><td>Friendly user interface</td></tr>
<tr><td>Related Requirements</td><td>Functional requirements for the videogame</td></tr>
<tr><td>Goal in Contex</td><td>To make a more approachable experience for the user as it plays the videogame</td></tr>
<tr><td>Preconditions</td><td>The website must load correctly
The website must load the game correctly</td></tr>
<tr><td>Successful end condition</td><td>The website will show the starting screen of the videogame.</td></tr>
<tr><td>Primary Actors</td><td>Server</td></tr>
<tr><td>Secondary Actor</td><td>User</td></tr>
<tr><td>Trigger</td><td>The user loads the website</td></tr>
<tr><td></td><td>
<table>
<tr><td>step</td><td>execution</td></tr>
<tr><td>1</td><td>The user enters the website</td></tr>
<tr><td>2</td><td>The server loads the website</td></tr>
<tr><td>3</td><td>The server loads the video game</td></tr>
<tr><td>4</td><td>The video game starts</td></tr>
<tr><td>5</td><td>The video game shows.</td></tr>
</table>
</td></tr>
</table>


User accesses the website
The server loads the game into the website
Server
User
The game interacts in the server
The interactions react correctly. The game moves correctly

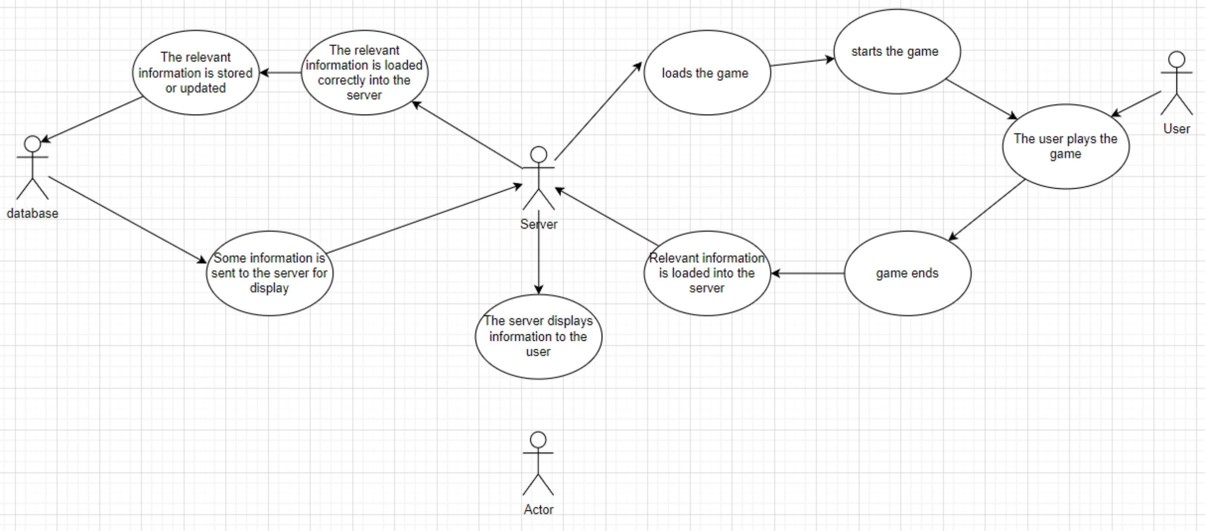
<table>
<tr><td>Use case name</td><td>Smooth gameplay user experience</td></tr>
<tr><td>Related Requirements</td><td>Those related to the game development</td></tr>
<tr><td>Goal in Contex</td><td>If a game has no positive outlook nor a positive impact then it is useless as a user experience</td></tr>
<tr><td>Preconditions</td><td>The game must be uploaded to the webpage and also must be functional</td></tr>
<tr><td>Successful end condition</td><td>The game will end smoothly without glitches</td></tr>
<tr><td>Primary Actors</td><td>Server</td></tr>
<tr><td>Secondary Actor</td><td>User</td></tr>
<tr><td>Trigger</td><td>The player loads the game and the start game screen appears</td></tr>
<tr><td></td><td><table><tr><td>Step</td><td>Execution</td></tr><tr><td>1</td><td>The game loads</td></tr></table></td></tr>
</table>

	2	The game initializes
	3	The game runs
	4	The game ends



Use case name	Data integrity remains after CRUD functions
Related Requirements	Setup MySQL queries Setup MySQL tables Add scoring script
Goal in Contex	To maintain data stability and prevent erroneous information with data integrity when inserting, deleting, updating or extracting from or to the database

<table>
<tr><td>Preconditions</td><td>Connection between game and website
Initialization of the game
Hardware for the server must be up and running
Data types and integrity restrictions must be specified</td></tr>
<tr><td>Successful end condition</td><td>Player data is inserted, deleted, updated or extracted with no errors, and valid data is displayed within the game or website.</td></tr>
<tr><td>Primary Actors</td><td>Server, developer</td></tr>
<tr><td>Secondary Actor</td><td>Player</td></tr>
<tr><td>Trigger</td><td></td></tr>
<tr><td></td><td>
<table>
<tr><th>Step</th><th>Execution</th></tr>
<tr><td>1</td><td>Player loads game</td></tr>
<tr><td>2</td><td>Player starts level</td></tr>
<tr><td>3</td><td>Player generates data while playing the game</td></tr>
<tr><td>4</td><td>Data is sent to the server</td></tr>
<tr><td>5</td><td>Data is stored in the server</td></tr>
<tr><td>6</td><td>Data is updated while the player generates more data</td></tr>
<tr><td>7</td><td>Player ends level</td></tr>
<tr><td>8</td><td>Data is displayed within the website</td></tr>
</table>
</td></tr>
</table>


The relevant information is stored or updated
The relevant information is loaded correctly into the server
loads the game
starts the game
User
The user plays the game
database
Server
Some information is sent to the server for display
The server displays information to the user
Relevant information is loaded into the server
game ends
Actor