INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

Campus Santa Fe

TC2005B - Software Development and Decision Making

Master Documentation for *Heaven't* as proposed by the *Mexican Association of Videogames*

Prof. Esteban Castillo, Prof. Gilberto Echeverria, Prof. Octavio Navarro

June 13th, 2022

Emiliano Cabrera, A01025453

Diego Corrales, A01781631

Andrew Dukerley, A01025076

Do Hyun Nam, A01025276

Abstract

Heaven't's development is mainly centered around the goal of using a videogame as an educational tool. The following document is written to summarize all the development processes revolving around *Heaven't*, from the ideation of the game, to its implementation on a WebGL interface.

Main objectives

The game has the following objectives in mind:

- The game allows the user/player to create their own playable maps, which will be stored in a database for others to play them as well.
- The database also stores information about the user (authentication), level, and stats recorded by each play by the end user.
- The API is intended to connect and serve the database to both UIs (game interface and website).
- A website is developed to show statistics of the levels created and played by the user. It also serves as a settings manager to manage the user's levels.

Scrum stats

All the software development was done according to the SCRUM framework. Where all team members acted at least once as the scrum master fulfilling every functional and non-functional requirements week by week, sprint by sprint.

<u>Functional requirements</u>

1. The system provides a User Interface that has build, statistics and play sections.
2. The system allows the user to create levels.
    a. The system allows the user to upload levels.
3. The system allows the user to play their own and others levels.
    a. The system allows users to exit a game.
    b. The system allows the user to move a character.
    c. The system allows the user to attack
    d. The system allows the user to deal damage upon using a melee or ranged attack.

    e. The system allows the user to change weapons.
    f. The system allows the user to recover health on a health item pickup.
    g. The system allows the user to take damage upon receiving a melee or ranged attack.
    h. The system allows the user to use spells.
    i. The system allows the user to change spells
    j. The system allows the user to end the game either by dying or defeating the boss.
4. The system stores user-created levels on a MySQL database.
    a. The system allows for level sharing using the database.
5. The system allows the user to authenticate from the browser.
    a. The system is capable of running in a web browser.
    b. The webpage is built using React.
    c. The backend is built using Node.
6. The system is built using the unity engine.
7. The API is built using Express.

<u>Non-functional requirements</u>

1. The system allows the user to set a level name.
2. The system allows the user to dash with a melee attack.
3. The system spawns monsters using a spawner.
4. The system allows the user to pause the game.
5. The system allows the user to recover faith on faith item pickup.
6. The system allows the user to dash and refresh set dash with a melee attack

<u>Sprints</u>

1. Week 1: 12 hours. Scrum master: Diego Corrales.
    a. Game
        i. Player can attack, take damage, move (5 hours).
        ii. Player can recover faith and HP on item pickup (5 hours).
    b. Database
        i. Database preliminary design (2 hours).

2. Week 2: 12 hours. Scrum master: Andrew Dunkerley.
    a. Game
        i. Player can end the game by dying or defeating the boss (5 hours).
        ii. Finding sprites that matched our concept over the open source (1 hour).
    b. Database
        i. Database relationships defined (1 hour).
        ii. Database schemas defined (2 hours).
        iii. Create v0.0.1 of the database for API testing (1 hour).
    c. API
        i. Create a FastAPI project and connect with the database (1 hour).

3. Week 3: 25 hours. Scrum master: Do Hyun Nam.
    a. Game
        i. Scripts defined for each object and prefab (6 hours).
            1. Monsters
            2. Player
            3. Walls
            4. Tilemap
        ii. User can create levels from the game (6 hours).
        iii. Player can switch between spells (3 hours).
        iv. Player can deal damage (3 hours).
            1. Player can use spells to attack .
            2. Player can use weapons.
    b. Database
        i. Established final database schemas with 3NF (1 hour).
        ii. Deploy database to a cloud droplet via a Docker container (1 hour).
    c. API
        i. Created endpoints to create, read, update and delete data for each database table (4 hours).
            1. Users table

2. Levels table
3. Level stats table

ii. Endpoint testing (1 hour).

d. Website

i. Initialized React project and set predefined styles (1 hour).

ii. Created log in and sign in page (1 hour).

iii. Users can authenticate using the website and see their data (2 hours).

4. Week 4: 15 hours. Scrum master: Emiliano Cabrera
    a. Game
        i. Users can upload their levels for them and others to play (7 hours).
        ii. Users can exit a game at will (1 hour).
        iii. The game can run in a web browser (1 hour).
    b. Website
        i. Users can scout levels created by other players from the frontend (2 hours).
        ii. Users can see the levels created by them (2 hours).
        iii. Basic stats are displayed in the level stats page (2 hours).
5. Week 5: 7 hours. Scrum master: Diego Corrales
    a. Game
        i. Fix details and deploy to the web (1 hour).
    b. Database
        i. Add three different views (1 hour).
        ii. Add triggers to update data from level stats to user and levels (1 hour).
        iii. Add stored procedures to execute comparisons between data rows (1 hour).
    c. API
        i. Add GET endpoints for stored procedures and views created (1 hour).
    d. Website
        i. Integrated WebGL build with the website, game is playable online (2 hours).

ii. Add graphs for the different views created in the DB (1 hour).

Working hours

Name	Time invested
Emiliano Cabrera	26 hours
Diego Corrales	26 hours
Andrew Dunkerley	27 hours
Do Hyun Nam	25 hours

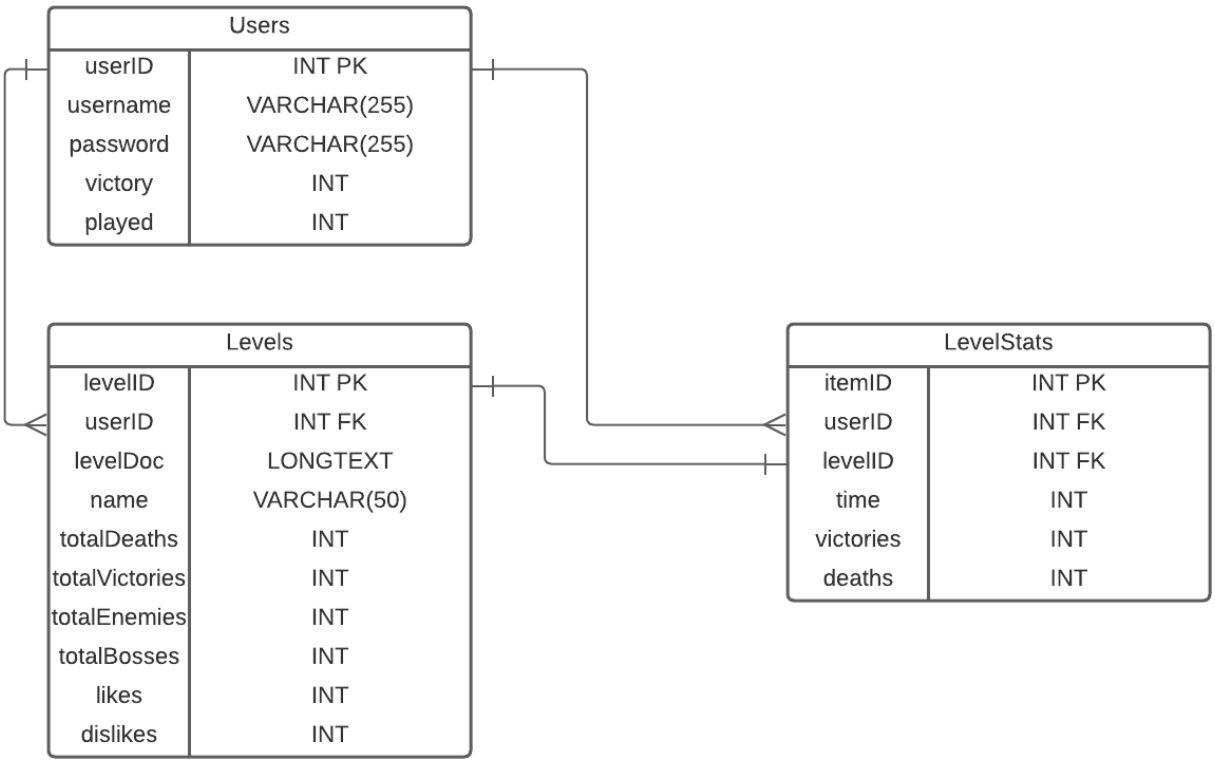
Database

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 and all values of each table are strictly dependent on only the set primary key.
    - This in its effect, prohibits functional dependencies inside of each table. In our case, we don't have that many tables and none of the attributes in any of our tables can be obtained through another value of another attribute.
- Each table has no complex keys, each primary key is one valued.
- Every value established and defined is atomic.
    - This means that all data needs to be atomized, all cells must contain at least a default value (meaning no *null* values). And not a single record can be equal to another. This means that in the case that a value is not provided, a default value that substitutes set value in the meantime is required. This behavior is reflected in our table through the command known as NOT NULL and in the cases that value may not be strictly required, DEFAULT can be implemented so that there's never an invalid record in the database with empty cells. At the same time, the command AUTO_INCREMENT is found in all of the ID definitions so that this field doesn't need to be filled up on upload.
- Our database script has anti redundancy measures that prohibit the same username from being created.

- There are no transitive tables in our database.
    - This means there is not a single 3rd attribute that is functionally dependent on a second attribute which in its own is dependent on a first attribute; in which case it would mean that the third attribute is dependent on the first attribute transitively through the second attribute. In our model every connection made between the tables is strictly to between those two tables alone meaning transitive connections do not exist in our model.
    -

Entity-relationship diagram



La base de datos diseñada cuenta con tres tablas:

- Users: Stores the data of each user, as well as their authentication methods
- Levels: It saves the statistics relevant to each level that is created within the game, as well as the information in a LONGTEXT type JSON to load the levels in the game engine.
- LevelStats: Saves individual stats for each user for each level they play.

Regarding data integrity, it was decided to define most of the variables in simple data types (INT and VARCHAR) to avoid reading complexities and errors when transforming the data received from the API. Certainly, in a more integrated development environment, it would be optimal to use more specific data types for each variable that is stored within the database.

User stories

User Story #1 Level Editor

As the president of the *Asociación Mexicana de Videojuegos*
I want the player to edit levels and its elements
to offer a tool for the creation of video-games.

Validation: ● Levels can be edited tile-by-tile. ● Additional elements (items, enemies, decorations) can be added to a level. ● The characteristics of enemies (HP, damage, speed) can be edited.	**Value:** 1000. **Priority:** 1. **Estimate:** 3 weeks.

User Story #2 Level Saving

As the president of the *Asociación Mexicana de Videojuegos*
I want the players to be able to save their levels locally
to have an accessible and permanent record of all created and saved levels by the player.

Validation: ● Have an option to save a created level in-game. ● Check that all the data of a created level is stored in a local database.	**Value:** 1000. **Priority:** 1. **Estimate:** 4 weeks.

User Story #3 Database Design

As a representative of the *Asociación Mexicana de Videojuegos*
I want the database utilized by the game to be relational in design and in third normal form
to be able to make modifications to the database without compromising the data in it with discrepancies or redundancies.

Validation:
- Check that the database contains multiple relations connected by foreign keys.
- Check that the database is in third normal form.
- Check that no register is repeated in the database.

Value: 200.

Priority: 2.

Estimate: 1 week.

User Story #4 Online Database

As the president of the *Asociación Mexicana de Videojuegos*
I want the database utilized by the game to be hosted in an online server
to be able to access the data remotely as a player or developer.

Validation:
- Check that the database is hosted in an online server.
- The online database contains all of the game's data.
- The online database can be accessed through any computer that has the required permissions.

Value: 500.

Priority: 2.

Estimate: 2 weeks.

User Story #5 Level Uploading

As the president of the *Asociación Mexicana de Videojuegos*
I want the players to be able to upload their levels to the online database
to have an accessible and permanent record of all created and uploaded levels by all users.

Validation:

- Have an option to upload a created level in-game.
- Check that all the data of a created and uploaded level is stored in the online database.

Value: 1000.

Priority: 1.

Estimate: 4 weeks.

User Story #6 Level Download

As the president of the *Asociación Mexicana de Videojuegos*
I want the players to be able to download levels uploaded by other players to the online database
to generate interaction between players and allow for the creation of more content by them.

Validation:

- There is a screen to download levels in-game.
- A player can find all levels uploaded to the database in-game.
- Selecting a level from the download screen downloads that level to their game.
- Downloaded levels can be edited or played.

Value: 500.

Priority: 2.

Estimate: 4 weeks.

User Story #7 Combat System

As the president of the *Asociación Mexicana de Videojuegos*
I want the gameplay of the game to include a basic combat system
to make the game fun and attractive to play.

Validation: ● The player can perform a basic attack. ● The player can use spells/abilities. ● The player can dash.	**Value:** 200. **Priority:** 3. **Estimate:** 1 week.

User Story #8 Weapon Variety

As representative of the *Asociación Mexicana de Videojuegos*
I want the game to offer a small variety of weapons
to make the game editing more interesting and allow for inventory progression.

Validation: ● There are a minimum of 3 different weapons in the game. ● There is a “light” weapon. ● There is a “heavy” weapon. ● There is a ranged weapon.	**Value:** 100. **Priority:** 4. **Estimate:** 1 week.

User Story #10 Objective

As representative of the *Asociación Mexicana de Videojuegos*
I want the player to pass a level y reaching a certain point
to motivate the player to keep playing.

Validation:
- A level ends once a certain part of a level is reached. This displays a victory screen.
- The player can edit where the level ending is.
- The player can die before reaching the objective if their HP reaches 0.

Value: 200.

Priority: 3.

Estimate: 1 week.

User Story #11 Monster Spawning

As the president of the *Asociación Mexicana de Videojuegos*
I want creators to be able to place a monster spawner the will place monsters periodically
to give levels some longevity and challenge

Validation:
- Enemy spawners can be placed instead of individual enemies
- Enemy parameters can be modified from the spawner itself

Value: 200.

Priority: 4.

Estimate: 1 week

User Story #12 Game Use Data

As the president of the *Asociación Mexicana de Videojuegos*
I want statistics on the use of the game by its players to be tracked
to learn useful information about their tendencies.

Validation: ● Statistics (time spent on a level, most frequently used enemies, percentage of people who passed the level) are actively tracked and recorded on the database.	**Value:** 500. **Priority:** 2. **Estimate:** 3 weeks.

User Story #13 Hosting the Game on a Webpage

As the president of the *Asociación Mexicana de Videojuegos*
I want the game to be available to play by anyone on a webpage
to make it easily accessible to all players.

Validation: ● Check that the game can be played on a webpage through any computer.	**Value:** 1000. **Priority:** 1. **Estimate:** 4 weeks.

User Story #14 Displaying Statistics

As the president of the *Asociación Mexicana de Videojuegos*
I want the tracked statistics on the use of the game to be displayed visually on the hosting webpage
to make the statistics easily accessible to any user.

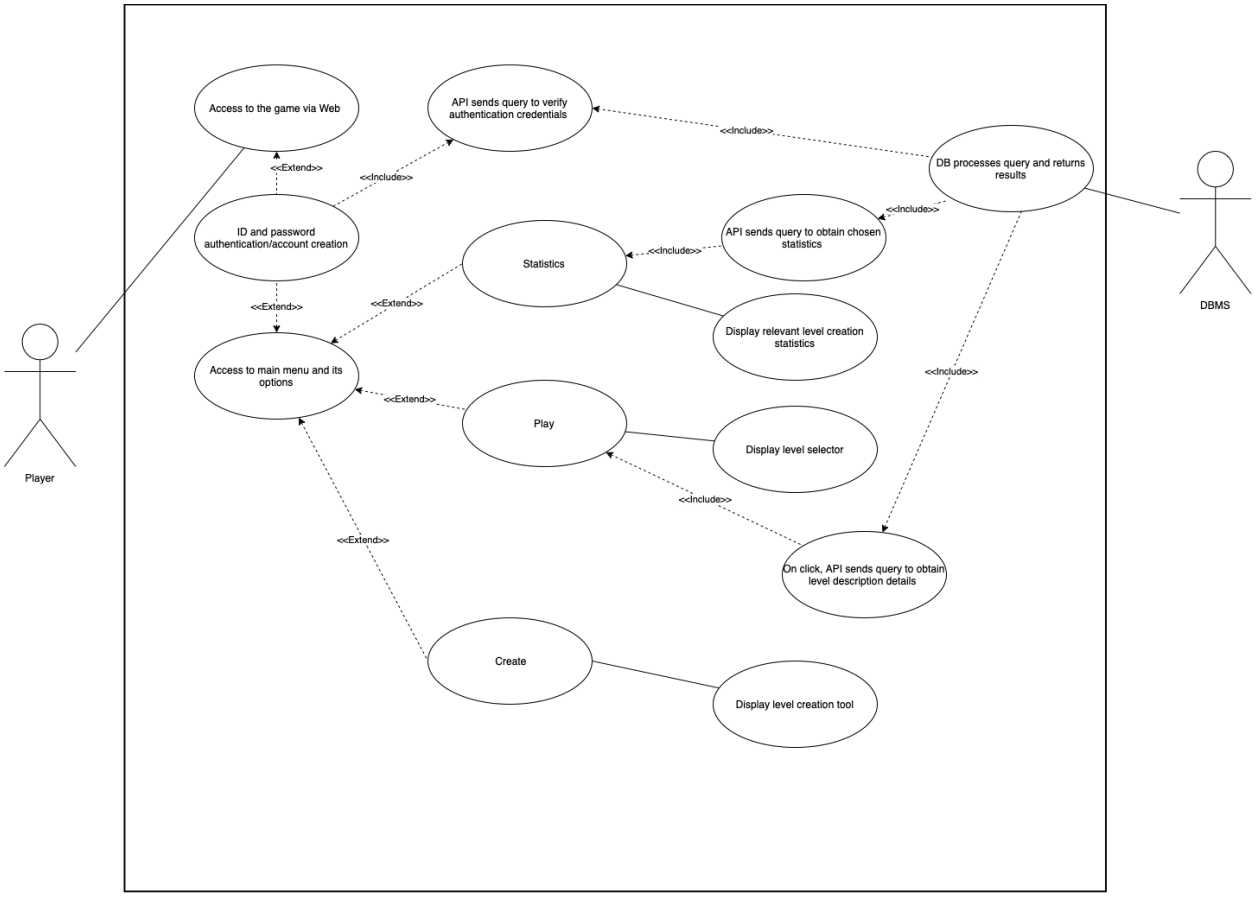
Validation: ● All tracked statistics are displayed in a visual and meaningful manner on the hosting webpage.	**Value:** 500. **Priority:** 2. **Estimate:** 2 weeks.

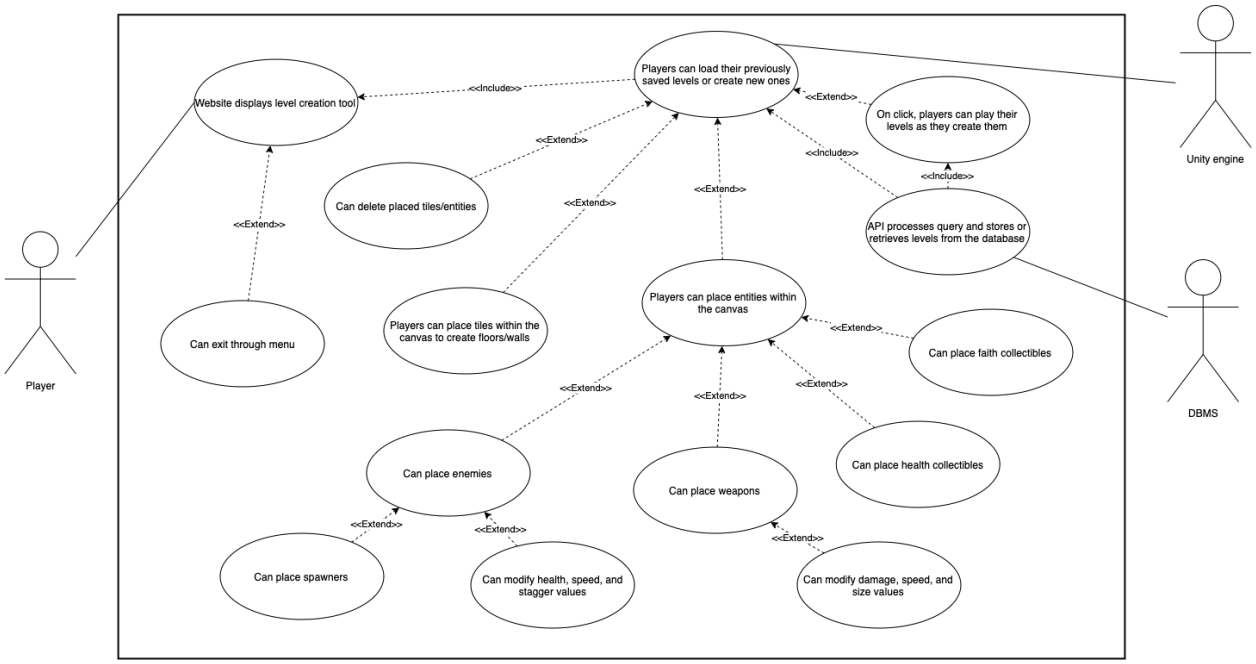
User Story #15 User Authentication

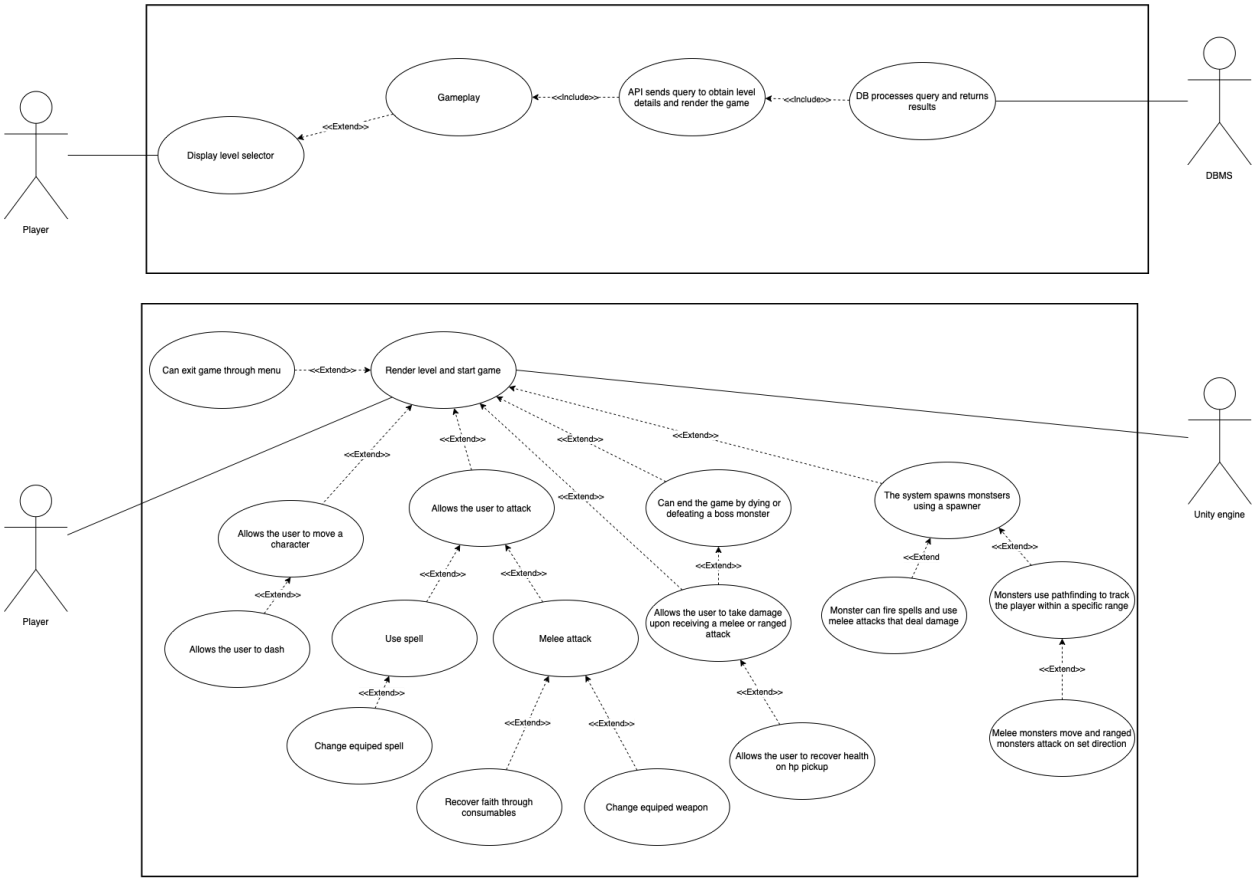
As the president of the *Asociación Mexicana de Videojuegos*
I want player information to be stored in relation to an account
to maximize security and track statistics

Validation: ● Players can log in using an email and password ● Information about the player is remembered across play sessions	**Value:** 1000. **Priority:** 1. **Estimate:** 3 weeks

UML diagrams







Use case tables

General Use Case Diagram									
Use case name	**Related Require-ments**	**Goal in Context**	**Preconditions**	**Successful end condition**	**Failed End Condition**	**1st Actors**	**2nd Actors**	**Trigger**	**Main Flow**
Access to main menu and its options	5,5b,5c, 7	A player enters the game and its main menu	The player has valid credentials	user exits the game	none	Player	none	User passes credential validation	Step 1: Browser loads the game's main menu
Play	1, 3	A player can enter the play menu	The player must select the play menu	user exits play menu	none	Player	none	User selects play menu	Step 1: Player selects the play button
Statistics	1, 5b	A player can enter	The player must select	user exits statistics	none	Player	none	User selects	Step 1: Player selects the statistics button

		the statistics menu	the statistics menu	menu				Statistics menu	
Create	1,2,4	A player can enter the create menu	The player must select the create menu	user exits create menu	none	Player	none	User selects Create menu	Step 1: Player selects the play button
Displays relevant level creation statistics	1,4	Display relevant statistics that are retrieved from the database	A player must be inside the statistics section	user exits statistics menu	none	Player	none	API sends request	include::API sends query to obtain statistics include::DB processes query and returns result Step 1: Browser prints relevant statistics table(s)
Display level selector	1,3	Display the different levels available to play	Player is located on website main menu	Player has selected a level	none	Player	none	Player clicks on level menu	include::API sends query to obtain level details include::DB processes query and returns result Step 1: Browser prints available levels
Display level creation tool	1,2,4	Display level creation tool	The player must be inside the create section	user exits play section or saves a level	none	Player	none	User selects Create menu	Step 1: Browser loads empty canvas with creation utility
Access to the game via Web	5a,5b	Enter the main menu via inputting webpage ip address or domain name	none	user closes the webpage	none	Player	none	connection request is made with the server	Step 1: Player inputs ip or domain name on the browser Step 2: hosting server grants permission
ID and password authentication/account creation	5	Verify Player credentials to enter the game	Access the portal via IP	user inputs correct credentials	user credentials are not correct or nonexistent	Player	none	Player submits credentials	Step 1: Player inputs id and password for either login or account creation include::API sends query to verify credentials include::DB

									processes query and returns result extend::A player enters the game and its main menu
API sends query to verify authentication credentials	4,7	To send input credentials to the db	Credentials must have been submitted	DB returns results	DB error	Player	none	API receives information	Step1: API receives and formats information into a query Step 2: API starts a connection with the DB Step 3: API sends query include::DB processes query and returns result Step 4: API returns the result
DB processes query and returns results	4	To process queries sent by the API	API must have established a connection and sent a query	DB processes the query and returns a result	DB error	Player	DBMS	API starts a connection	Step 1: DBMS engine processes the query Step 2: Returns a result
API sends query to obtain statistics	7	To send relevant statistics to the game	A request for the credentials must have been received	DB returns results	DB error	Player	none	API receives request	Step 1: API receives request Step 2: API starts connection with the DB include::DB processes query and returns result Step 3: Display relevant level creation statistics
API sends query to obtain level details	7	To send information about the playable levels in the game	A request for all levels must have been received	DB returns results	DB error	Player	none	API receives request	Step1: API receives request Step 2: API starts connection with the DB include::DB processes query and returns result Step 3: Display all available levels

Play Use Case Diagram									
Use case name	**Related Require-ments**	**Goal in Context**	**Preconditions**	**Successful end condition**	**Failed End Condition**	**1st Actors**	**2nd Actors**	**Trigger**	**Main Flow**
Can exit game through menu	3a	Game can be quitted at will by the player	Player is on gameplay	Player has clicked the designated button	none	Player	none	Player clicks ESC or designated key	Step 1: Player opens pause menu Step 2: Player selects quit option Step 3: Access to main menu and its options
Display level selector	2, 3	Display the different levels available to play	Player is located on website main menu	Player has selected a level	none	Player	none	Player clicks on level menu	include::API sends query to obtain level details include::DB processes query and returns result Step 1: Browser prints available levels
Allows the user to move a character	3m	Player can move freely through the game map	Player sprite is rendered on screen	Player isn't clicking a movement button	none	Player	none	Player clicks movement keys	Step 1: Player presses one of the 4 movement keys Step 2: Game engine processes inputs Step 3: Character moves for as long as the button is pressed
Allows the user to dash	3m, 3g	Player can speed up their movement	Player is moving with direction and at normal speed	Player isn't clicking dash or cooldown is 0	cooldown is not 0	Player	none	Player clicks dash key	Step 1: Player presses space bar Step 2: Game engine processes inputs Step 3: Character dashes set direction Step 4: start dash cooldown
On level click render level and start game	3	Player can play the desired level and start gameplay	Player is located on level selector	Player successfully clears level or dies	Levels are non-existent	Player	none	Player clicks START GAME	Step 1: Player clicks on desired level include:: API sends query to obtain level details and render game include:: DB

									processes query and returns results Step 2: Game engine renders level
Allow the user to attack	3d	Player can deal damage to monsters	Player is on gameplay	Player has missed or dealt the attack	none	Player	none	Player clicks attack key	Step 1: Player presses attack key Step 2: Game engine processes input Step 3: Player attacks
Use spell	3e	Player uses ranged attacks to deal damage	Player is on gameplay and has enough faith	Player has not enough faith, missed the attack, or dealt the attack	none	Player	none	Player has attack over spell and attack key clicked	Step 1: Player has attack type of range Step 2: Player presses attack key extend:: Allows user to attack
Change equiped spell	3b	Player can select the type of ranged attack	Player is on gameplay and has enough faith	Player has chosen a different spell	Player selects same spell	Player	none	Spell change function key	Step 1: Player presses spell hotkey Step 2: The type of spell changes extend:: use spell
Melee attack	3d	Player uses melee attacks to deal damage	Player is on gameplay and monster at close range	Player has missed or dealt the attack	none	Player	none	Player has attack over melee and attack enabled	Step 1: Player has attack type of weapon Step 2: Game engine processes input extend:: Allows user to attack
Recover faith through melee attacks	3k	Player recharges energy to continue ranged attacks	Player is on gameplay and has successfully dealt damage	Player has missed or dealt the attack	Faith is full	Player	none	Player successfully hits monster	extend:: Melee attack Step 1: Player has successfully attacked
Change equiped weapon	3f	Player can select the type of melee attacks through a weapon	Player is on gameplay	Player has chosen a different weapon	Player selects same weapon	Player	none	Weapon change function key	Step 1: Player presses weapon's hotkey Step 2: The type of weapon changes extend:: Allows user to attack
API sends query to obtain level	4, 4a	Game connects to API	Levels are stored over the database	Query returns data successfull	There are no levels stored	DBMS	none	GET request	include:: DB processes query and returns results

details and render game		layer to retrieve and display a level		y or error					Step 1: The API receives a GET request Step 2: The API sends a query to DBMS include:: DB processes query and returns results Step 3: The API receives the sent data or error
Can end the game by dying or deleting a boss monster	3c	Methods to end gameplay by the player	Player is on gameplay and has defeated all mobs	Player has died or killed the boss	none	Player	none	Player dies or eliminates boss	extend:: Allows the user to take damage upon receiving attack Step 1: Game logic checks player or boss health Step 2.: If player health is 0 player dies, same logic for boss Step 3: If one of them has a health of 0, game ends
Allows the user to take damage upon receiving a melee or ranged attack	3i	Player loses hit points based on the type and value of attack	Player is on range for the monster's hitbox	Player has been damaged successfully	Player is dead	Player	none	Monster hits player	Step 1: Monster attacks user successfully Step 2: Health is reduced based on the power of the attack
Allows the user to recover health on HP Pickup	3l	Player gains hit points with certain items	Player doesn't have full health	Player has picked up a healing item	Player has full health	Player	none	Player picks health item	Step 1: Game logic checks player health is not full Step 2: Player steps a health item
DB processes query and returns results	4	The API layer sends query to retrieve the needed data to the database	DB has levels stored within	Query returns data successfully or error	Query returns data successfully or error	Player	DBMS	Query submitted by API	Step 1: API send query Step 2: DBMS processes the query Step 3: DBMS sends results or error

The system spawns monsters using a spawner	3h	Monsters are created from certain spawn points defined by spawner object	Spawners are placed and time elapsed	A monster is placed on the level successfully	Number of monsters reach spawn limits	Player	none	Cooldown reaches 0	Step 1: Game logic checks that monster limit is not reached Step 2: Game logic checks timer for spawning is 0 Step 3: Spawns monster
Monster can fire spells and use melee attacks that deal damage	2c, 3j	Monsters use different types of attacks to deal damage to the player	Monster is spawned using spawner	Monster dealt or missed attack	none	Player	none	Monster has player in range	extends:: The system spawns monsters using a spawner Step 1: Player is in monster's range Step 2: Monster attacks towards player direction
Monsters use pathfinding to track the player within a specific range	2d	Monsters fix attacks to user when player enters a radiobox	Player is within the monster's hitbox	Monster has sensed a player close to it	Player is dead	Player	none	Monster has player in range	extends:: The system spawns monsters using a spawner Step 1: Player is in monster's range Step 2: Monster moves towards player vector
Melee monsters move and range monsters attack on set direction	3j	Monsters move towards the player when on proximity	Monster has detected a player in its hitbox	Monster has a player's direction	none	Player	none	Monster has player in range	extends:: Monster use pathfinding to track the player within a specific extends:: The system spawns monsters using a spawner range extends:: Monster can fire spells and use melee attacks that deal damage

Create Use Case Diagram									
Use case name	**Related Require-ments**	**Goal in Context**	**Preconditions**	**Successful end condition**	**Failed End Conditio**	**1st Actors**	**2nd Actors**	**Trigger**	**Main Flow**

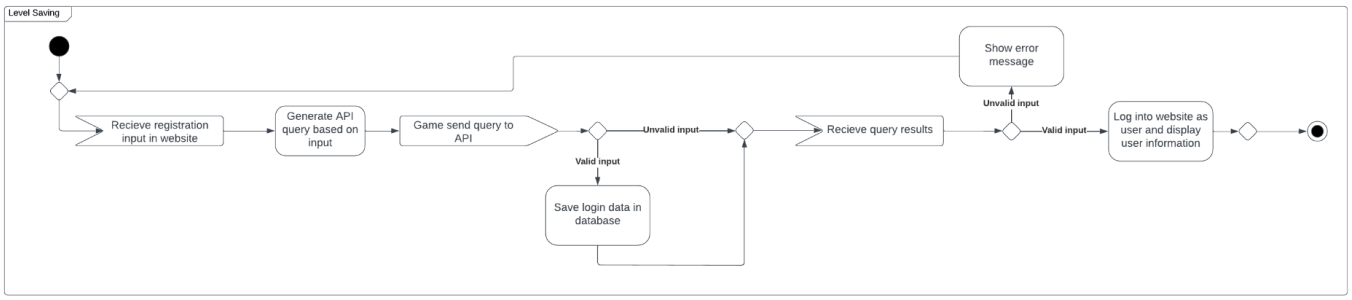
					n				
Website dipsplays level creation tool	2	A player has access to the creation tools	The player must select the create option in the main menu	user exits the game	user cannot exit the game	Player	none	User chooses Create option in main menu	Step 1: Players selects CREATE option in main menu
Players can load their previously saved levels or create new ones	2a, 4,4a	A player can make levels or keep modifying prior ones	The player must have saved levels or space to make them	user can retrieve a level from the database	user cannot retrieve levels from the database	Player	DBMS	User chooses to make a new level/save d level	include:: Website dipsplays level creation tool Step 1: Player chooses Load level option Step 2: Browser loads player's levels Step 3: Player can choose any of their saved levels
On click, players can play their levels as they create them	4, 4a	A player can be their own tester	The player must be creating/mo difying a level	user can stop playing their level	user cannot play their level	Player	none	User chooses to play within the creation screen	extend:: Players can load their previously saved levels or create new ones Step 1: Player chooses Play level option Step 2: Browser loads game view with player's level
API processes query and stores or retrieves levels from the database	4, 4a	A player can store their level creator progress	The player must be able to pass a level	user can store a level in the database	user cannot save levels in the database	Player	DBMS	User chooses to save their progress	include:: Players can load their previously saved levels or create new ones include:: On click, players can play their levels as they create them Step 1: Player chooses to save level Step 2: Browser converts level into storable file Step 3: API inserts level file into the database
Can exit	3a	A player	The player	user exits the	user	Player	none	User	extend:: Website

through menu		can stop creating a level	must be creating/modifying a level	create screen	cannot exit the create screen			chooses to leave create screen	displays level creation tool Step 1: Player selects exit in menu
Can delete placed tiles/entities	2	A player can remove undesired items from the canvas	The player must have placed tiles/entities within a level	user removes tiles or entities from the canvas	user cannot remove items form the canvas	Player	none	User chooses to delete items	extend:: Players can load their previously saved levels or create new ones Step 1: Player chooses Delete option Step 2: Player chooses tile/entity to delete Step 3: Tile/Entity is deleted from the canvas
Players can place tiles within the canvas to create floors/walls	2	A player can define a playable area	The player must be within the create screen	user can place tiles within the canvas	none	Player	none	User chooses a tile to place	extend:: Players can load their previously saved levels or create new ones Step 1: Player chooses tile to place within the canvas Step 2: Player places a tile
Players can place entities within the canvas	2	A player can add dynamic objects to the level	The player must be within the create screen	user can add entities to the canvas	none	Player	none	User chooses an entity to place	extend:: Players can load their previously saved levels or create new ones Step 1: Player chooses entity to place within the canvas Step 2: Player places an entity
Can place enemies	2, 2c	A player can add enemies to their level	Step 1: player loads page and enters the main menu	user can add enemies to the canvas	none	Player	none	User chooses an enemy to place	extend:: Players can place entities within the canvas Step 1: Player chooses enemy to place Step 2: Player places an enemy
Can place spawners	2, 3h	A player can add	The player must be	user can add spawners to	none	Player	none	User chooses a	extend:: Can place enemies

		enemy spawners to their level	within the create screen	the canvas				spawner to place	Step 1: Player chooses a spawner Step 2: Player places a spawner within a canvas Step 3: Player chooses enemy
Can modify health, speed, and stagger values	2, 2d	A player can customize enemy behavior and danger	The player must be within a monster's values screen	user can modify enemies	none	Player	none	User chooses an enemy to modify	extend:: Can place enemies Step 1: Player chooses an enemy Step 2: Browser displays modifiable values Step 3: Player modifies values
Can place weapons	2	A player can add weaponry to their level	The player must be within the create screen	user can add weapons/spells to the canvas	none	Player	none	User chooses a weapon to place	extend:: Players can place entities within the canvas Step 1: Player chooses a weapon/spell to place Step 2: Player places a weapon/spell
Can modify damage, speed, and size values	2, 3e, 3f	A player can modify weapon effectiveness	The player must be within a weapon's 7/spell's values screen	user can modify weapons/spells	none	Player	none	User chooses a weapon to modify	extend:: Can place weapons Step 1: Player chooses a weapon Step 2: Browser displays modifiable values Step 3: Player modifies values
Can place health collectibles	2, 3l	A player can make healing possible within their level	The player must be within the create screen	user can add healing drops to the canvas	none	Player	none	User chooses to place healing	extend:: Players can place entities within the canvas Step 1: Player chooses health drops Step 2: Player places a health drop
Can place faith collectibles	2	A player can give ammunition to other players	The player must be within the create screen	user can add faith drops to the canvas	none	Player	none	User chooses to place faith	extend:: Players can place entities within the canvas Step 1: Player chooses faith drops

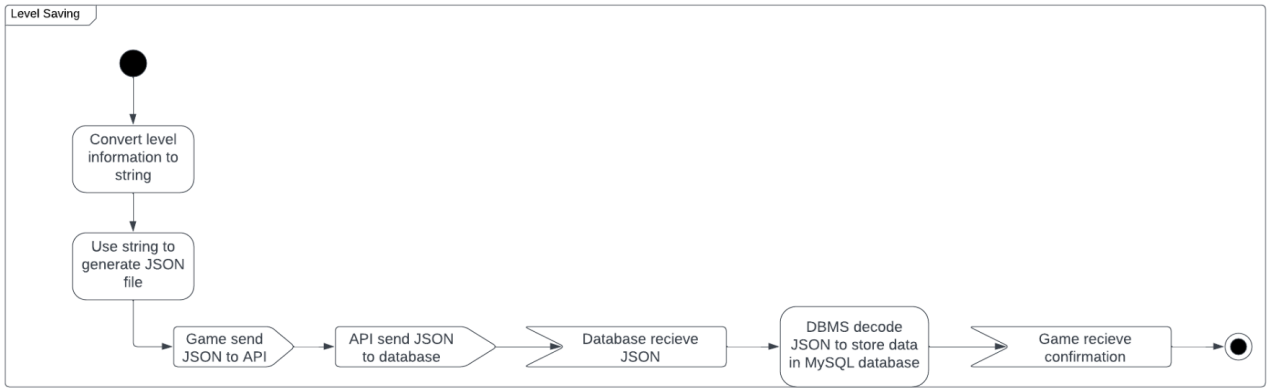
		within their level							Step 2: Player places a faith drop

UML activity diagrams



Related requirements:

5.The system allows for user authentication within the web browser



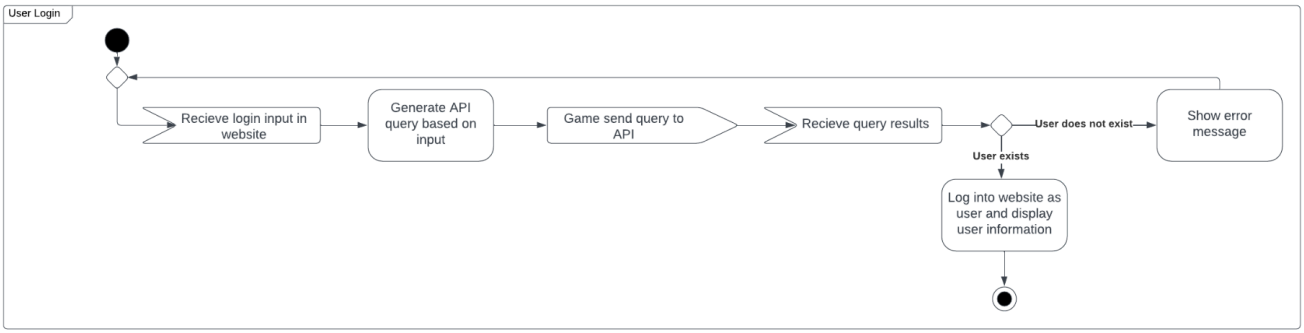
Related requirements:

4.The system stores user-created levels on a database built with MySQL

4a.The system allows for level sharing using the aforementioned database

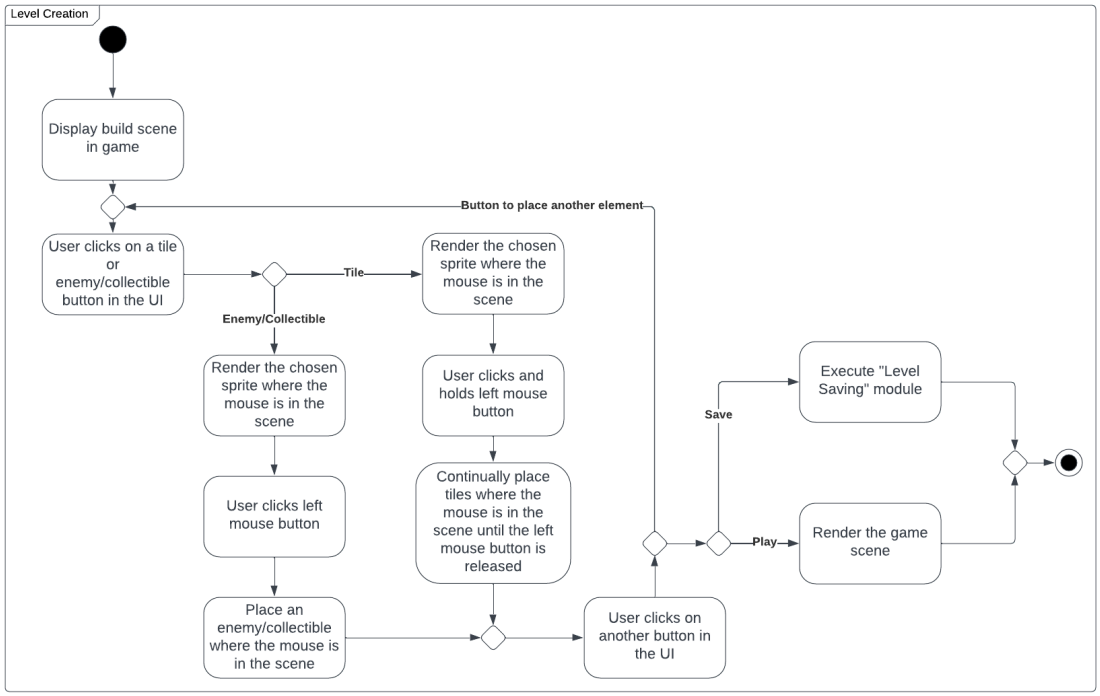
Related requirements:

3.The system allows the user to play their own and others users' levels



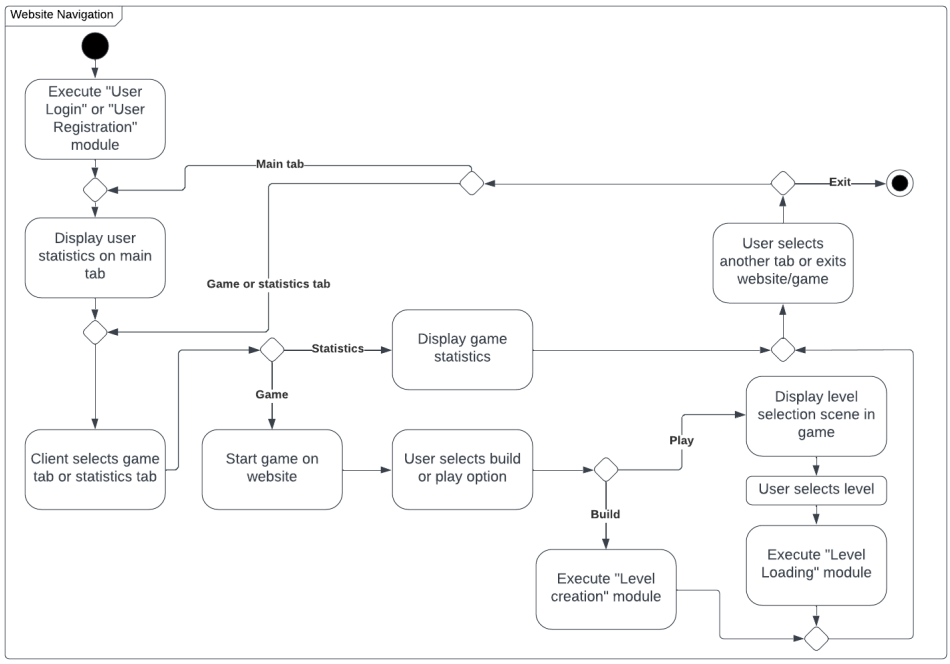
Related requirements:

5.The system allows for user authentication within the web browser



Related requirements:

2.The system allows the user to create levels



Related requirements:

1.The system provides a UI that has build, statistics and play sections

2.The system allows the user to create levels

3a.The system allows the user to exit a game

5a.The system is capable of running on a web browser