SOFTWARE ENGINEERING PROBLEM SPECIFICATION TABLE

Davide Flamini Cazaran - A00381665

CLIENT	Video game company
USER	Developer and administrator
FUNCTIONAL REQUIREMENTS	- R1: Register level - R2: Create a player - R3: Register enemy at a level - R4: Register treasure at a level - R5: Modify the score of a player - R6: Increase the level of a player - R7: Report the treasures and enemies of a level given by the user - R8: Report the amount found of a treasure in all levels - R9: Report the amount found of a type of enemy in all levels - R10: Report the most repeated treasure in all levels. - R11: Inform the enemy that gives the highest score and the level where it is located. - R12: Report the number of consonants found in the names of the enemies in the game. - R13: Inform the top 5 of the players according to the score.
PROBLEM CONTEXT	- The game has a menu with 12 options - The maximum number of treasures in the game is 50 - The maximum number of players in the game is 20 - There can only be a maximum of 25 enemies in - the game The game has 10 levels - The menu it will be executed according to the option entered by the user
NON-FUNCTIONAL REQUIREMENTS	- RN1: Compatibility: That it works in a web application and in a mobile app. - RN2: Performance: In the deployment of the treasures, in the web application it should not take more than 2 seconds.

Functional requirements analysis table

<table>
<tr><td>Name or identifier</td><td colspan="3">R1: Register level</td></tr>
<tr><td>Summary</td><td colspan="3">The system must allow registering the different levels of the game. From a level you have a number that identifies it, the points that are required to go to the next level.

The level of complexity (high, medium, low), is defined if the points awarded by the treasures is greater than the points of the enemies of the level, it is low level, if it is the same, it is medium, and if the points of the enemies are more. enemies that the points of the treasures are high.</td></tr>
<tr><td rowspan="5">Inputs</td><td>Input name</td><td>Data type</td><td>Selection or repetition condition</td></tr>
<tr><td>idNum</td><td>int</td><td></td></tr>
<tr><td>pointToNextLevel</td><td>int</td><td>That the user has entered a valid data type in the previous variable</td></tr>
<tr><td>sumPointsEnemies</td><td>int</td><td></td></tr>
<tr><td>sumPointsTreasures</td><td>int</td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Receive the level creation parameters
2. Check if there is space in the
3. level array Add the level to the array of levels of the game
4. Calculate the difficulty level from the comparison of the sum of the points of the treasures, with the sum of the points of the enemies
5. Returns a boolean value, depending on whether the operation could be performed</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">The boolean condition that confirms whether the level could be added</td></tr>
<tr><td rowspan="2">Outputs</td><td>Name Output</td><td>Data type</td><td>Selection or repetition condition</td></tr>
<tr><td>resultLevel</td><td>boolean</td><td>Than inputs and the method processes have worked correctly</td></tr>
</table>

Name or identifier	R2: Create a player		
Summary	The system must allow players to register, they have a nickname that identifies them fica, a name, the initial score (the player starts with 10), has a number of lives (starts with 5). The nickname cannot be repeated.		
Inputs	Input name	Data type	Selection or repetition condition
	nickname	String	
	name	String	That the entered nickname is not repeated
General activities necessary to obtain the results	1. Receive the player creation parameters 2. Check that the nickname is not repeated 3. Check if there is space in the array players 4. Add the player to the system players array 5. Returns a boolean value, depending on whether the operation could be performed		
Result or postcondition	The boolean condition that confirms whether the player can be added		
Outputs	Name Output	Data type	Selection or repetition condition
	resultAddPlayer	boolean	Than inputs and the method processes have worked correctly

Name or identifier	R3: Register enemy		
Summary	The system must allow registering enemies at different levels of the game. In a level the enemies cannot be repeated because once defeated, you would already know how to defeat the others of the same level. The possible types of enemies in the game are: ogres, abstract, boss and magical. A position X and a position Y must be generated randomly. Taking into account the resolution of the game.		
Inputs	Input name	Data type	Selection or repetition condition
	nameE	String	
	typeE	enum	That the user has entered a valid data type in nameE

	damagePoints	int	
	pointsE	int	That the user has entered a valid data in the previous variable
General activities necessary to obtain the results	1. Receive the parameters of the creation of the enemy 2. Check if there is space in the array of enemies 3. Check that the enemy is not repeated 4. Create a random X and Y position. These are attributes of the enemy 5. Add the enemy to the array of enemies if there is space 6. Returns a boolean value, depending on whether the operation could be performed		
Result or postcondition	The boolean condition that confirms whether the enemy could be added to the level		
Outputs	Name Output	Data type	Selection or repetition condition
	resultAddEnemy	boolean	Than inputs and the method processes have worked correctly

Name or identifier	R4: Register treasures		
Summary	The system must allow registering treasures at different levels of the game. In a level the same treasure can be found in different positions, that is, a diamond can be found in two different positions in the same level, so when entering the treasures into the game, the user will be asked how many treasures will be registered for a same level. A position X and a position Y must be generated randomly. Taking into account the resolution of the game.		
Inputs	Input name	Data type	Selection or repetition condition
	nameT	String	
	image	String	That the user has entered a valid data type in nameT
	pointsT	int	That the user has entered a valid data in the previous variable

General activities necessary to obtain the results	1. Receive the parameters of treasure creation 2. Check if there is space in the treasure array 3. Create a random X and Y position. These are attributes of the treasure 4. Add the treasure to the treasure array if there is space 5. Returns a boolean value, depending on whether the operation could be performed		
Result or postcondition	The boolean condition that confirms whether the treasure could be added to the level		
Outputs	Name Output	Data type	Selection or repetition condition
	resultAddTreasure	boolean	Than inputs and the method processes have worked correctly

Name or identifier	R5: Modify a player's score		
Summary	The system must allow to modify a player's score. It should be remembered that the player starts with 10 points.		
Inputs	Input name	Data type	Selection or repetition condition
	nickname	String	
	getPoints	int	That the user has entered a nickname that exists
	newPoints	int	
General activities necessary to obtain the results	1. Receive the parameters for modifying a player's score 2. Check if the player's nickname exists 3. Modify the player's score, by the new score 4. Returns a confirmation message		
Result or postcondition	A confirmation message is displayed as to whether the operation could be carried out correctly		
Outputs	Name Output	Data type	Selection or repetition condition
	resultChangePointsPlayer	String	Than inputs and the method processes have worked correctly

<table>
<tr><td>Name or identifier</td><td colspan="3">R6: Increase level for a player</td></tr>
<tr><td>Summary</td><td colspan="3">The system must allow increase level for a player, in case you cannot increase the level, you must inform the user what score is required to increase.</td></tr>
<tr><td rowspan="4">Inputs</td><td>Input name</td><td>Data type</td><td>Selection or repetition condition</td></tr>
<tr><td>nickname</td><td>String</td><td></td></tr>
<tr><td>getPoints</td><td>int</td><td>That the user has entered a nickname that exists</td></tr>
<tr><td>getPointToNextLevel</td><td>int</td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Receive the parameters for a player's level increase
2. Check if the player's nickname exists
3. Verify if the player's points are equal to or greater than the number of points to go to the next level
4. If the player meets condition 3, level up the player
5. Returns a confirmation message. If the level up failed, the message will show what score you need to level up.</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">A confirmation message is displayed as to whether the operation was successful.</td></tr>
<tr><td rowspan="2">Outputs</td><td>Name Output</td><td>Data type</td><td>Selection or repetition condition</td></tr>
<tr><td>resultLevelUp</td><td>String</td><td>Than inputs and the method processes have worked correctly</td></tr>
</table>

<table>
<tr><td>Name or identifier</td><td colspan="3">R7: Report the treasures and enemies of a given level</td></tr>
<tr><td>Summary</td><td colspan="3">The system must report the treasures and enemies (separated by commas) of a level given by the user.</td></tr>
<tr><td rowspan="4">Inputs</td><td>Input name</td><td>Data type</td><td>Selection or repetition condition</td></tr>
<tr><td>idNum</td><td>String</td><td></td></tr>
<tr><td>getTreasure</td><td>Treasure</td><td>That the user has entered a level that exists</td></tr>
<tr><td>getEnemy</td><td>Enemy</td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Receive the parameters to show the treasures and enemies of a given level
2. Check if the level exists
3. Explore the arrangements of treasures and enemies and in the positions that are not empty get their attributes</td></tr>
</table>

	4. Returns a message where the information is displayed		
Result or postcondition	The list of enemies and treasures of a given level is displayed on the screen		
Outputs	Name Output	Data type	Selection or repetition condition
	msj	String	Than inputs and the method processes have worked correctly

Name or identifier	R8: Report the amount of treasure found at all levels		
Summary	The system must report the amount of treasure found at all levels, that is, if the user wants to know how many diamonds exist in all levels.		
Inputs	Input name	Data type	Selection or repetition condition
	nameTreasureToSearch	String	
	getLevel[]	Level array	That has put the name of a treasure that exists
	getNameT	String	
	getTreasure[]	Treasure array	
General activities necessary to obtain the results	1. Receive the parameters to show the amount of a treasure in all levels 2. Check that the treasure exists 3. In each level treasure array compare the name entered by the user, with the existing names in the arrays 4. If the comparison of the name is the same, add 1 to a counter 5. Returns a message where the information is displayed		
Result or postcondition	The number of times the treasure appears in all levels is displayed on the screen		
Outputs	Name Output	Data type	Selection or repetition condition
	msj	String	Than inputs and the method processes have worked correctly

Name or identifier	R9: Report the amount found of an enemy in all levels
Summary	The system must report the c found amount of a type of enemy in all levels, that is, if the user wants to know how many ogres exist in all levels.

Inputs	Input name	Data type	Selection or repetition condition
	typeEnemyToSearch	enum	
	getLevel[]	Level array	That the data chosen in the enumeration is correct
	getTypeE	String	
	getEnemy[]	Enemy array	
General activities necessary to obtain the results	1. Receive the parameters to show the amount of an enemy in all levels 2. Check that the type of enemy is correct 6. In each array of enemies in the levels, compare the type entered by the user, with the existing types in the arrays 7. If the comparison of the type is the same, add 1 to a counter 8. Returns a message showing the information		
Result or postcondition	The number of times an enemy type appears on all levels is shown on the screen		
Outputs	Name Output	Data type	Selection or repetition condition
	msj	String	Than inputs and the method processes have worked correctly

Name or identifier	R10: Inform the most repeated treasure in all levels		
Summary	The system must inform which and It's the most repeated treasure in all levels.		
Inputs	Input name	Data type	Selection or repetition condition
	getTreasure[]	Treasure array	
	getLevel[]	Level array	
	getNameT	String	

General activities necessary to obtain the results	1. Receive the parameters to show the most repeated treasure at all levels 2. A counter associated with each name of treasure 3. In each arrangement of treasures in the levels, count how many times the same name of a treasure 4. is repeated Each time the name is repeated, increase the counter of each treasure by 1 5. Check which is the name of the treasure that appears the most 6. Returns a message where the information is displayed		
Result or postcondition	The number of times and the name of the treasure that appears the most in all levels is displayed on the screen		
Outputs	Name Output	Data type	Selection or repetition condition
	msj	String	Than inputs and the method processes have worked correctly

Name or identifier	R11: Inform the enemy that gives the highest score and the level where it is located		
Summary	The system must inform which is the enemy that gives the highest score and the level where it is located.		
Inputs	Input name	Data type	Selection or repetition condition
	getEnemy[]	Enemy array	
	getLevel[]	Level array	
	getTypeE	enum	
	getPointsE	int	
General activities necessary to obtain the results	1. Receive the parameters to show the enemy that gives the highest score 2. In each array of enemies in the levels, by using a conditional, compare which enemy has the poinstE variable that has the largest value 3. After having made the total route, save the enemy and the level where it is 4. Return a message where the information is displayed		
Result or postcondition	It is displayed on screen the enemy that gives the highest score and the level where it is located.		
Outputs	Name Output	Data type	Selection or repetition condition
	msj	String	Than inputs and the method processes have worked correctly

Name or identifier	R12: Report the number of consonants found in the names of the enemies in the game		
Summary	The system must report the number of consonants found in the names of the enemies in the game		
Inputs	Input name	Data type	Selection or repetition condition
	getEnemy[]	Enemy array	
	getLevel[]	Level array	
	getNameE	String	
General activities necessary to obtain the results	1. Receive the parameters to display the number of consonants in the names of the enemies 2. A counter is created 3. In each level treasure array, count how many times a consonant is found 4. Each time a consonant is found, increase the counter by 1 5. returns a message showing the information		
Result or postcondition	It is shown on the screen in the number of consonants found in the names of the enemies of the game		
Outputs	Name Output	Data type	Selection or repetition condition
	msj	String	Than inputs and the method processes have worked correctly

Name or identifier	R13: Inform the top 5 of the players according to the score.		
Summary	The system must show the 5 players with the highest score within the game		
Inputs	Input name	Data type	Selection or repetition condition
	getPlayer[]	Player array	
	getPoints	int	

<table>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Receive the parameters to show the top 5 players
2. In the arrangement of players, compare their scores by means of a conditional
3. The 5 players with the highest score will be saved
5. A message is returned showing the information</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">. The top 5 players with the highest score are displayed on the screen. The attributes of each player will be displayed.</td></tr>
<tr><td rowspan="2">Outputs</td><td>Name Output</td><td>Data type</td><td>Selection or repetition condition</td></tr>
<tr><td>msj</td><td>String</td><td>Than inputs and the method processes have worked correctly</td></tr>
</table>