REQUIREMENTS SPECIFICATION

Client:	Icesi University
User:	People who want to play the Blitz Bomb game.
Functional Requirements	FR1. User registration FR2. Menú FR3. Init-Gameplay FR4. Graph generation FR5. Time calculation FR6. Avatar movement FR7. Dijkstra power-up FR8. Save Score
Problem Context	We aim to implement a software application that provides an engaging gaming experience. To achieve this, we will develop a game called "Blitz Bomb." The game should be action-based, where players take on the role of Kelvin, an alien character, and navigate through a complex network of interconnected bombs in the city of Cali. The main objective of the game is to allow players to strategically activate all hidden bombs while optimizing their path through the network of gunpowder pathways to reach the exit before time runs out. The game should deliver thrilling gameplay by incorporating elements of graph theory and graph algorithms for pathfinding. Additionally, "Blitz Bomb" should offer multiple difficulty levels, where the game dynamically calculates and adjusts time constraints for the player based on the chosen level.

Non-functional requirements	NFR1. The user interface must be intuitive and user-friendly, catering to players of experience levels. NFR2. The game must be developed using JavaFX as the primary technology for building the user interface and graphical components. NFR3. The game must incorporate two different graph implementations, namely an adjacency matrix data structure and an adjacency list data structure. These two graph implementations should be interchangeable without affecting the game's functionality.

Name and identifier	*[FR1 - User registration]*		
Summary	*The software should allow users to register in the game by providing a nickname. This registration is necessary to assign a score based on the time completed in each level and to save the player's progress.*		
Inputs	**Input Name**	**Data Type**	**Valid Value Conditions**
	nickName	String	Characters (text), no void
Result or postcondition	*After registration, the system will create a player profile associated with the provided nickname. This profile will include information about the player, such as the name, and will be empty in terms of scores and game progress.*		
Outputs	**Output Name**	**Data Type**	**Format**
	confirmation	String	Characters (text)

Name and identifier	*[FR2 - Menu]*		
Summary	*The software should display a menu to the user that includes the following options: "Play," "Difficulty," and "Graph Type." When the user selects the "Difficulty" option, three additional choices become available: "Easy," "Medium," and "Hard." If the user chooses the "Graph Type" option, two additional options are revealed: "Adjacency List" and "Matrix."*		
Inputs	**Input Name**	**Data Type**	**Valid Value Conditions**
	play	ActionEvent	N/A
	difficult	MouseEvent	Must be easy level, medium level or hard level
	graphType	MouseEvent	Must be: with matrix or with adjacency list.
Result or postcondition	Once the user selects the "play" option, the game will progress with the chosen difficulty level and graph type. In cases where the player has not specified a difficulty or graph type, the default settings will be "easy" for difficulty and "with matrix" for graph type. Subsequently, the game proceeds with the initial gameplay.		
Outputs	**Output Name**	**Data Type**	**Format**
	msg	String	Characters(Text)

<table>
<tr><td>Name and identifier</td><td colspan="3">[FR3 – Init-Gameplay]</td></tr>
<tr><td>Summary</td><td colspan="3">The software should allow the user to view the game by displaying a randomly generated graph. It will also present the estimated time using an animated panel that functions as a countdown timer, with the feature that the time decreases in real-time. Additionally, it will display a high score table containing the scores(user's time) of individuals who have played the game.

The software will position the player's avatar at the initial vertex, which will be marked in green. The player's objective is to traverse all the vertices containing bombs and reach the final vertex, also marked in green, before the designated time runs out. If the player reaches the final vertex without activating all the bombs, they will lose the game.</td></tr>
<tr><td rowspan="2">Inputs</td><td>Input Name</td><td>Data Type</td><td>Valid Value Conditions</td></tr>
<tr><td>N/A</td><td>N/A</td><td>N/A</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">The software will display the game to the user. This includes presenting the randomly generated graph, the estimated time via an animated panel functioning as a real-time countdown timer, and the visualization of the high-score table containing player scores. Additionally, the player's avatar will be positioned at the initial vertex marked in green, and the objective is to traverse all vertices containing bombs and reach the final vertex also marked in green before the designated time expires. If the player reaches the final vertex without activating all the bombs, the game is considered lost.</td></tr>
<tr><td rowspan="2">Outputs</td><td>Output Name</td><td>Data Type</td><td>Format</td></tr>
<tr><td>game</td><td>Scene</td><td>N/A</td></tr>
</table>

<table>
<tr><td>Name and identifier</td><td colspan="3">[FR4 - Graph generation]</td></tr>
<tr><td>Summary</td><td colspan="3">The software should allow the user to select the graph implementation they wish to use, whether it be via an adjacency list or an adjacency matrix.

Once the play option has been selected, the software will automatically generate a random simple graph consisting of a minimum of 50 vertices, each vertex of the graph shall have at most 4 edges. Some of these vertices will contain bombs, which will be visually represented by bomb images on the corresponding nodes. On the other hand, vertices without bombs will display an empty space.

The graph should be visually presented to the player in such a way that the edges between the nodes are represented by images of a gunpowder path, enhancing immersion in the game's theme. Furthermore, the software should display the weight of each edge to the user. This will enable the player to clearly identify both the bomb locations and the connections between nodes.

Additionally, it must be ensured that the generated graph is connected, signifying that all nodes will be interconnected in some way, and there will be no isolated nodes. This ensures gameplay consistency and the possibility of completing the levels.</td></tr>
<tr><td rowspan="2">Inputs</td><td>Input Name</td><td>Data Type</td><td>Valid Value Conditions</td></tr>
<tr><td>N/A</td><td>N/A</td><td>N/A</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">The system will generate a graph for the game. The graphs may vary in size and layout to provide a unique experience in each playthrough.</td></tr>
<tr><td rowspan="2">Outputs</td><td>Output Name</td><td>Data Type</td><td>Format</td></tr>
<tr><td>graphList</td><td>N/A</td><td>N/A</td></tr>
</table>

<table>
<tr><td>Name and identifier</td><td colspan="3">[FR5 - Time calculation]</td></tr>
<tr><td>Summary</td><td colspan="3">The software will use graph algorithms to calculate the time in which the user will have to run the graph. For each game, different algorithms will be used: Prim and DFS.

The Prim algorithm will be employed to find the minimum spanning tree of the graph. In other words, it will seek a subset of edges that form a graph with all vertices, minimizing the total weight of the edges. This minimum spanning tree will serve as a guide for the most efficient path the player should follow to traverse the graph.

The DFS algorithm will be used to traverse the minimum spanning tree and sum the weights of all the edges. Each edge weight represents the time required to move from one node to the next. The sum of these weights will be used to calculate the estimated time the player should take to complete the graph.

The time allotted to the player to complete the level will vary depending on the selected difficulty level (easy, medium, difficult). In the difficult level, the assigned time will be calculated by the software using the Prim and DFS algorithms. In the medium level, an additional 10 seconds will be granted compared to the difficult level, and in the easy level, an additional 15 seconds will be provided compared to the difficult level. This allows for an adjustment of the challenge based on the player's preferences.</td></tr>
<tr><td>Inputs</td><td>Input Name</td><td>Data Type</td><td>Valid Value Conditions</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">The software uses the relevant algorithms to calculate the minimum time in which the user should traverse the graph, taking into account the "difficult" level, and it calculates the time for the chosen difficulty.</td></tr>
<tr><td rowspan="2">Outputs</td><td>Output Name</td><td>Data Type</td><td>Format</td></tr>
<tr><td>seconds</td><td>int[]</td><td>integers</td></tr>
</table>

Name and identifier	*[FR6 – Avatar movement]*		
Summary	*To facilitate movement of the avatar, the software should allow the user to navigate the avatar from one vertex to another using the following keys: A (left), D (right), S (down), W (up).*		
Inputs	**Input Name**	**Data Type**	**Valid Value Conditions**
	up	KeyEvent	N/A
	down	KeyEvent	N/A
	left	KeyEvent	N/A
	right	KeyEvent	N/A
Result or postcondition	*The avatar position will be modified, the avatar will move to the vertex indicated by the key pressed by the user.*		
Outputs	**Output Name**	**Data Type**	**Format**
	N/A	N/A	N/A

<table>
<tr><td>Name and identifier</td><td colspan="3">[FR7- Dijkstra power-up]</td></tr>
<tr><td>Summary</td><td colspan="3">The software must enable the player to use a power-up based on the Dijkstra algorithm. This power-up can be employed by the player to determine the quickest route between two vertices, to do this the software must allow the user to select 2 vertices of the graph. It becomes particularly effective when the player aims to reach the exit without activating all the bombs. However, for each unexploded bomb, the player will incur a penalty of 30 seconds as a consequence. This feature adds strategic depth to the gameplay, allowing players to balance risk and reward in their approach.</td></tr>
<tr><td rowspan="3">Inputs</td><td>Input Name</td><td>Data Type</td><td>Valid Value Conditions</td></tr>
<tr><td>firstVertex</td><td>MouseEvent</td><td>N/A</td></tr>
<tr><td>secondVertex</td><td>MouseEvent</td><td>N/A</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">The fastest path will be calculated and shown to the player by changing the color of the edges that are part of the path. Si el jugador opta por salir sin activar todas las bombas, incurrirá en una penalización de tiempo adicional.</td></tr>
<tr><td rowspan="2">Outputs</td><td>Output Name</td><td>Data Type</td><td>Format</td></tr>
<tr><td>powerUp</td><td>Scene</td><td>N/A</td></tr>
</table>

Name and identifier	*[FR8 - Save Score]*		
Summary	*The system must be able to store player scores in a CSV file using the serialization process. This file will contain two separate columns: one for player names and another for the time it took them to complete the game.*		
Inputs	**Input Name**	**Data Type**	**Valid Value Conditions**
	N/A	N/A	N/A
Result or postcondition	*Whenever a user finishes a game, the system should automatically save their score in the respective file.*		
Outputs	**Output Name**	**Data Type**	**Format**
	scoreTable	CSV	CSV format