The goal of this project is to create a captivating and immersive maze game, leveraging graph-based structures, that provides an entertaining gaming experience while exploring the eerie and mysterious theme of the "**backrooms**."

Project Objective:

Our main objective is to design and implement a maze game with a "backrooms" theme that engages players in a thrilling adventure. This maze game will serve as a practical application of graph theory and algorithms. It will encompass various elements, from maze generation to player interaction, to create a seamless gaming experience.

1. Components and Functionalities (Maze Game - "Backrooms" Edition):

1.1. Maze Generation

We will employ graph structures to procedurally generate the maze. Each room in the maze will be represented as a vertex, and the connections between rooms as edges. This approach ensures that the maze is intricate and labyrinthine, capturing the essence of the "backrooms."

1.2. User Interface

A user-friendly interface is a critical component of our game. It will allow players to navigate, explore, and interact with the maze seamlessly. The interface will be designed to evoke the eerie and mysterious atmosphere of the "backrooms."

1.3. Exploration and Challenges

Players will embark on a journey within the maze, facing challenges such as puzzles, riddles, and unexpected encounters. The game will offer an immersive experience, encouraging players to solve challenges to progress further through the maze.

1.4. Objective and Escape

The main objective for the players is to find an escape route from the maze. This goal provides players with direction and purpose and drives their exploration and decision-making within the game.

1.5. Graph implementation

We will implement the maze structure using graph representations. Two versions, based on adjacency matrices and adjacency lists, will be created to ensure flexibility. These representations will be the backbone of the maze and will help in efficient pathfinding and exploration.

1.6. Graph Algorithms

To enhance the gaming experience, we will integrate graph algorithms such as depth-first search (DFS) and breadth-first search (BFS) for pathfinding and exploration within the maze. These algorithms will play a key role in guiding players through the complex network of rooms.

Name and ID	**R1: Generate Labyrinth**		
Review	*The system must generate the labyrinth in which the game will take place*		
Inputs	**Input name**	**Data type**	**Valid values**
	mazeConfiguration	Configuration object	Define maze parameters
Result	The labyrinth is generated		
Outputs	**Output name**	**Data type**	**Format**
	labyrinth	Graph	Representation of maze

Name and ID	**R2**: Verify possible paths to the exit.		
Review	*The system must verify and store the possible paths leading from the start to the exit using some search algorithm, either DFS or BFS.*		
Inputs	**Input name**	**Data type**	**Valid values**
	mazeGraph	Graph	*Representation of the labyrinth*
Result	All the possible paths are stored in a linked list composed of others linked list of vertexes.		
Outputs	**Output name**	**Data type**	**Format**
	paths	LinkedList	Linked list of linked lists

Name and ID	**R3:** Verify shortest path.		
Review	*The system should check and save the shortest path leading from the start to the exit, using a minimum weight paths algorithm such as Djikstra or Floyd-Warshall.*		
Inputs	**Input name**	**Data type**	**Valid values**
	mazeGraph	Graph	*Representation of the labyrinth*
Result	The shortest path is stored in a linked list of vertexes		
Outputs	**Output name**	**Data type**	**Format**
	shortestPath	LinkedList	Linked list of vertices

Name and ID	R4: Move character.		
Review	*The system must move the character in the maze whenever the user clicks on any of the available squares.*		
Inputs	**Input name**	**Data type**	**Valid values**
	userClick	Event	User clic position
Result	The character is moved to the new position.		
Outputs	**Output name**	**Data type**	**Format**
	characterMovement		Update character's position in the maze

Name and ID	R5: Calculate and show score		
Review	*The system must calculate the score of the player based on the number of steps taken and the treasures that the player picked up.*		
Inputs	**Input name**	**Data type**	**Valid values**
	numberOfSteps	int	Should be greater than 0
	numberOf Treasures	int	Should be greater than or equal to 0
Result	The score is calculated and shown to the user		
Outputs	**Output name**	**Data type**	**Format**
	score	String	Message with the score

Name and ID	R6: Treasures generation.		
Review	*The system must generate treasures in random locations.*		
Inputs	**Input name**	**Data type**	**Valid values**
	treasureConfig	Configuration object	Define parameters for treasure generation
Result	Treasures are generated in random locations.		
Outputs	**Output name**	**Data type**	**Format**
	treasure	Treasure	An object of the Treasure class in a random location.

Name and ID	R8: User interface (GUI).		
Review	*The system must deploy a user interface that allows the development of the game. Once the user chooses to start the game, the system must change the window to the labyrinth one, allowing the user to interact directly with the labyrinth and to see his score once the game is finished.*		
Inputs	**Input name**	**Data type**	**Valid values**
			-
Result	A window with the main menu and the button to start the game is shown.		
Outputs	**Output name**	**Data type**	**Format**
	mainWindow		A window showing the main menu and the respective button to start the game.

Name and ID	R7: Player inventory.		
Review	*The system must allow the user to view his inventory, which stores the treasures he has collected.*		
Inputs	**Input name**	**Data type**	**Valid values**
	userClick	Event	-
Result	The inventory is shown.		
Outputs	**Output name**	**Data type**	**Format**
	inventory	-	A window showing a table with the name treasures.