DAT602 – Assessment 1

Assignment 1 – Database Driven prototype game application

Alex Herbert

27/07/2024

## Introduction

This project is to design and implement a “point and click” game application that allows two or more players to play against each other at the same time from different devices via a network. Players turns are to be stored in the database in such a way that the database can be used to implement real-time gameplay. All rules, updates of moves, kills, picking up of items, or similar inventory functions during game play are to be implemented using TSQL procedures and functions. The main core functionality of the game application will be controlled using the database and the GUI is to merely provide the users with a platform to view the game and perform actions. The system that is to be implemented must support multiple copies of the game to be ran simultaneously. When a player is logged into their user account, the system will display a list of other online players and their highest score. Play is started by selecting an opponent that is currently not in a game, send a request, once both parties have accepted, they will start a new game.

## Game Concept

**Game Title:** Mouse Trap

Players control a mouse that must collect wedges of cheese that are scattered throughout the map, the map will be a basic maze grid, so players must navigate the maze to find the cheese. Along the way there will be mouse traps that spawn in random locations, these will block pathways so the player will either back track and try a different path, risk triggering the trap (this will have a chance of killing the player or taking of a large portion of health) or use an item that destroys the mouse trap clearing the path. The items that can spawn will be paper clips, which are used to trigger mouse traps to clear paths, and peanuts that can be used to regain health. If a player is killed by a trap, they will respawn at the starting tile of the map and will have to start again, any items/cheese they had collected before dying will be removed and redistributed into new tiles across the map.

The winner of the game will be the player that collects the most amount of cheese and makes their way to the centre of the map symbolised as a Mouse House.

## Game mechanics and rules

**Players and Movement**

* Players control a mouse character and navigate a maze to collect wedges of cheese.
* Players can move in four directions, up, down, left, right.

**Items and Obstacles**

* **Cheese**: Collected to score points.
* **Mouse Traps**: Randomly spawned to block pathways on the map. Can be destroyed with paper clips or the player can risk triggering the trap which may kill the mouse or deal a significant amount of damage.
* **Paper Clips:** Collect these items to be able to disable mouse traps.
* **Peanuts:** Collect to regain health points.

**Respawn Mechanics**

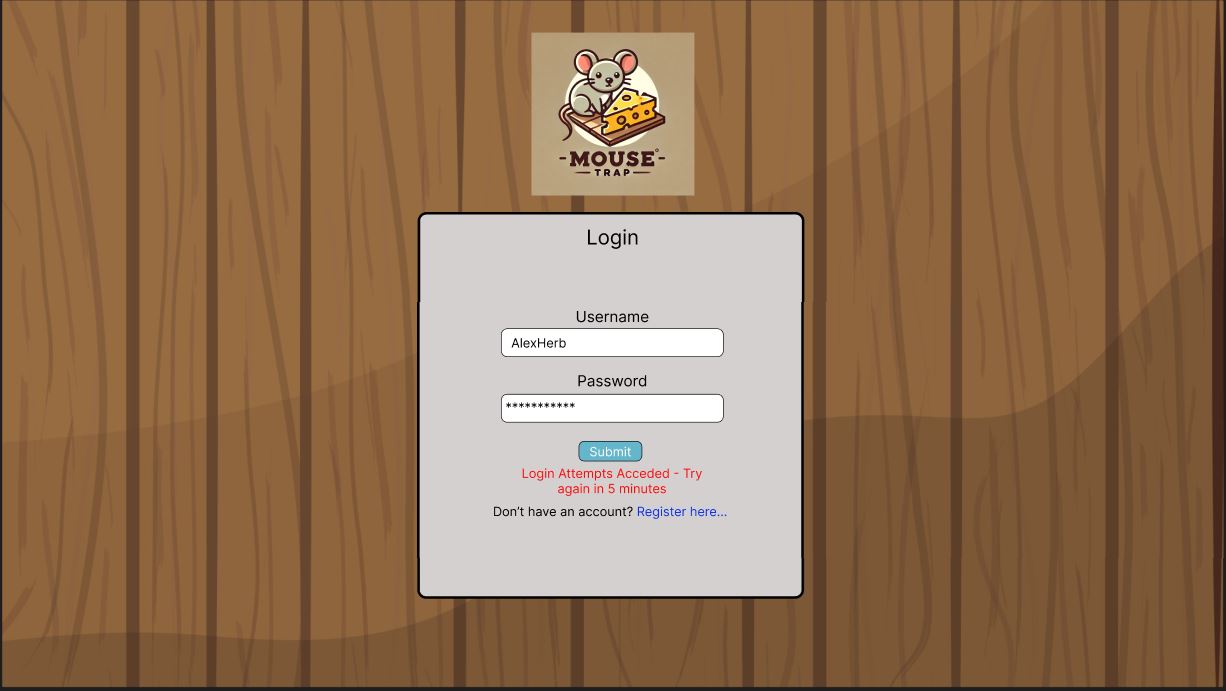
* On death, due to a mouse trap, players respawn at the starting tile, all their progress is reset and the items they have collected are redistributed around the map.

**Winning Condition**

* Collect the most cheese and reach the centre of the map to win the game. Score is saved and stored on the players account if it is a new high score.

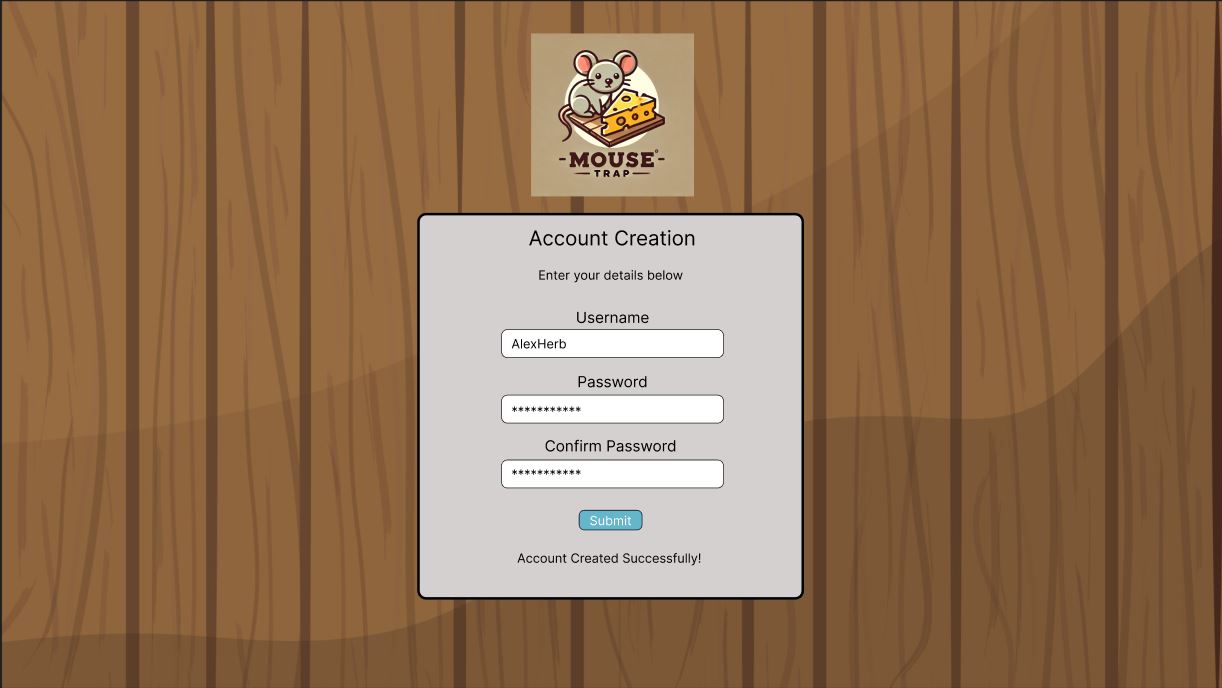
## StoryBoarding and screen design

### User Story 1 – Player opens the game, attempts to log in, gets locked out for 5 minutes



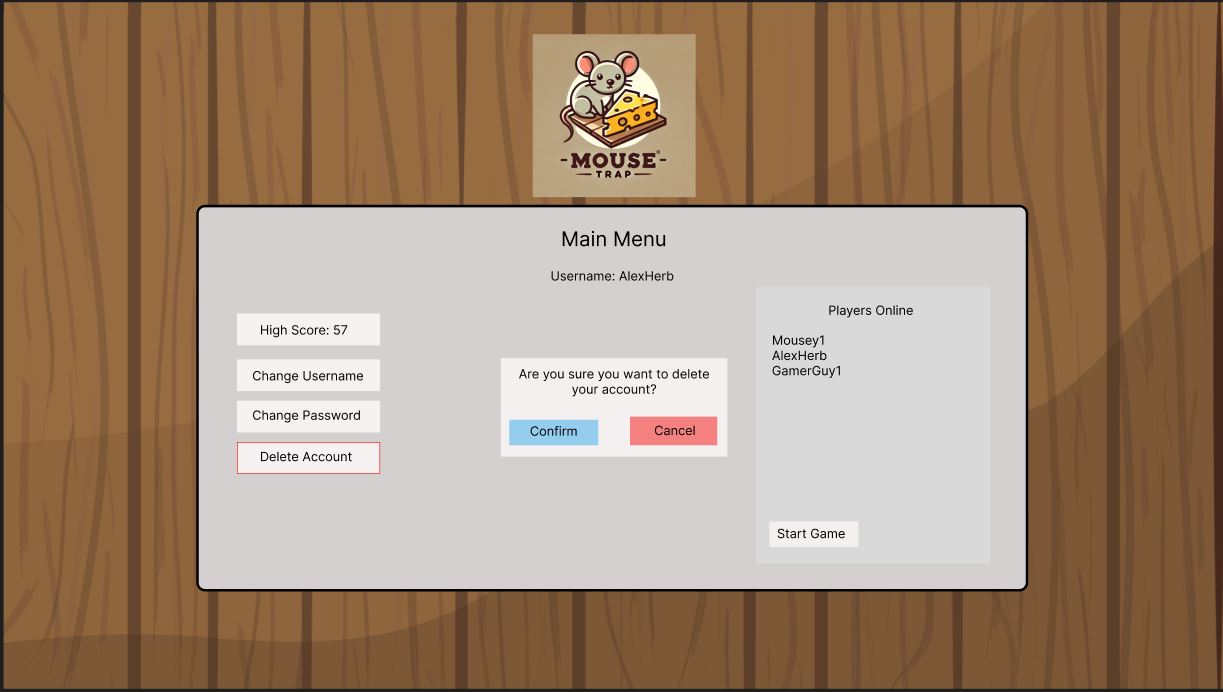
The player enters their log in credentials and attempts to log into their account using the log in form, after 3 unsuccessful attempts they are restricted from logging in for 5 minutes. After the 5 minutes is over, they enter the correct credentials and can log in to their account.

### User Story 2 – Player registration, Enters username and password, creates account



The player selected the “Register here…” link on the log in page and is presented with the registration form to create an account. They enter their username, password, and confirms their password before submitting the form. This then creates an account and logs the user in and redirects them to the main screen.

### User Story 3 – Player Deletes Own account using the Main Menu Options



Whilst logged into their account and on the main menu, the player has options to edit and update their account information as well as delete the account. The user selects the “Delete Account” button and is presented with a message asking them to confirm the action. The user selects “Confirm” and their account is deleted, and they are logged out and redirected to the Log in screen.

### User Story 4 – Game is started, tiles are placed on the game board

A screenshot of a game

Description automatically generated

When a game is started, the game board is displayed, and the tiles are placed in the correct positions on the grid. The “Barriers”, “Mouse House” and “Home Tile” of the maze are placed in fixed positions that will not change.

### User Story 5 – Placing Items on a tile



Once the Game Board has been loaded and all the tiles have been place, Items like “Cheese”, “Paper Clips” and “Mouse Traps” are placed on randomized tiles throughout the maze. Player Characters are also placed on the “Home Tile” to initiate the start of the game.

### User Story 6 – Player Game Movement



Once the game has been initiated, the player clicks on a tile that is adjacent to the “Mouse” player character, the player character is then moved to that tile and the players position is updated in the database.

### user Story 7 – Player gaining points and loosing points



The player gains points by moving to a tile that contains a “Cheese”, this updates there score with +1 once they have moved to that tile. The player loses points by moving to a tile that contains a “Mouse Trap”, if they do not have a “Paper Clip” to deactivate the trap then they will be hit by the trap and they may die, if the player dies they will lose all their points and the “Cheese” they have collected will be redistributed on the game board in random locations.

### User Story 8 – Player acquiring items and adding it to their inventory



To collect “Paper Clips’ for deactivation of traps, the player moves to a tile containing a “Paper Clip” item and the item will be added to the player’s inventory. “Peanuts” can be collected to restore health, the player moves to the tile containing “Peanuts” and their health is restored to maximum points.

### User Story 9 – Item (npc effect) movement

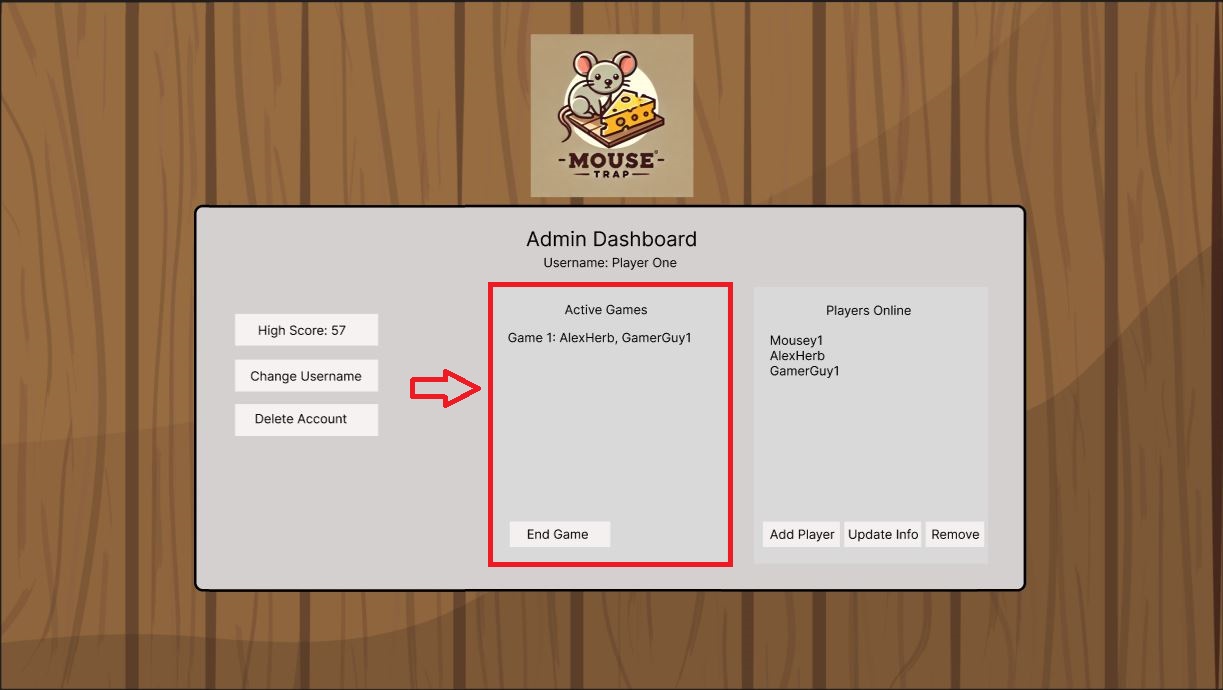
When a player deactivates or sets off a “Mouse Trap” the item is removed from the tile is was placed on, this is then placed on another tile at random on the game board.

### User Story 10 – Item Location



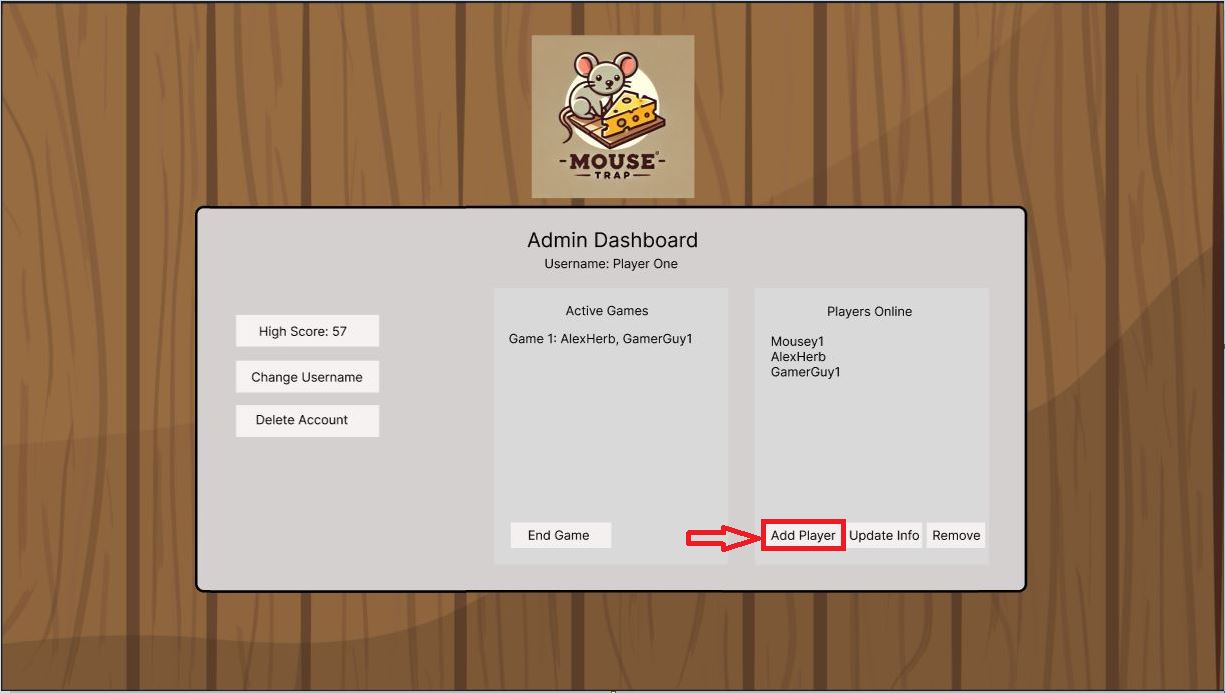
Item locations of the game board are randomly placed at the start and throughout the game loop, each item cannot be within 1 tile of another, each item will have empty tiles adjacent to them in all directions.

### User Story 11 – Game Administration, Kill running games



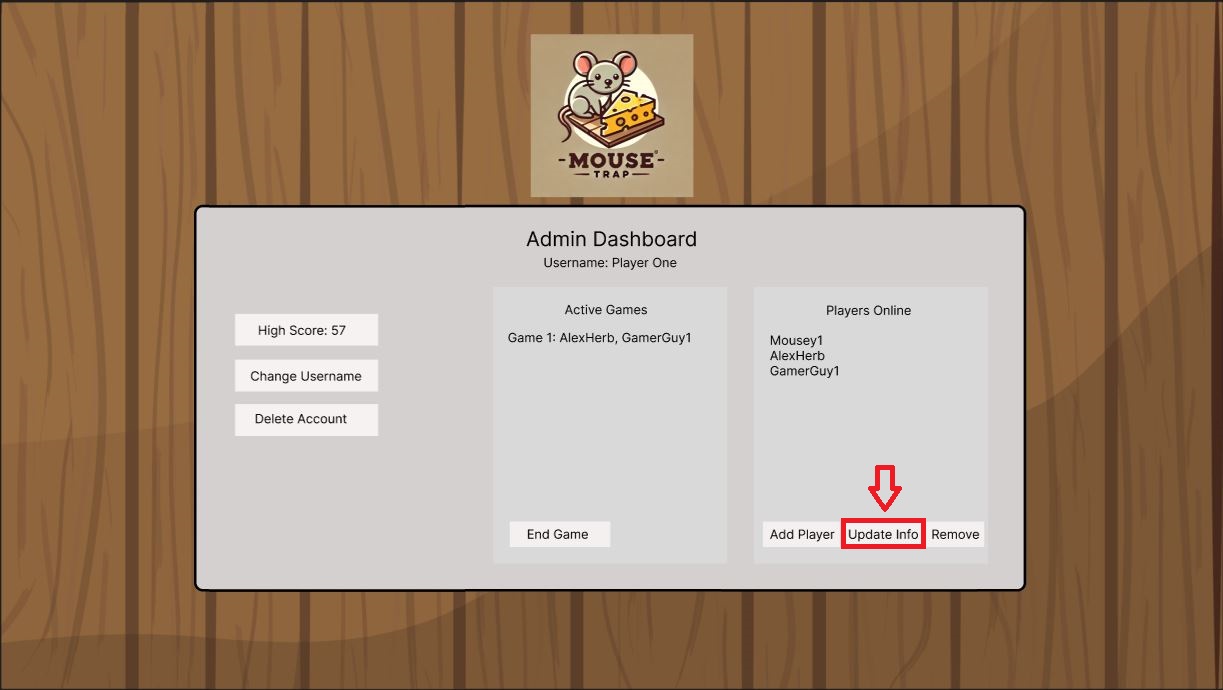
The Administrator user has access to the admin dashboard when logged in, they select an active game from the list, select the “End Game” button in the menu and the game is shut down and ended.

### User Story 12 – Game Administration, Add new Players



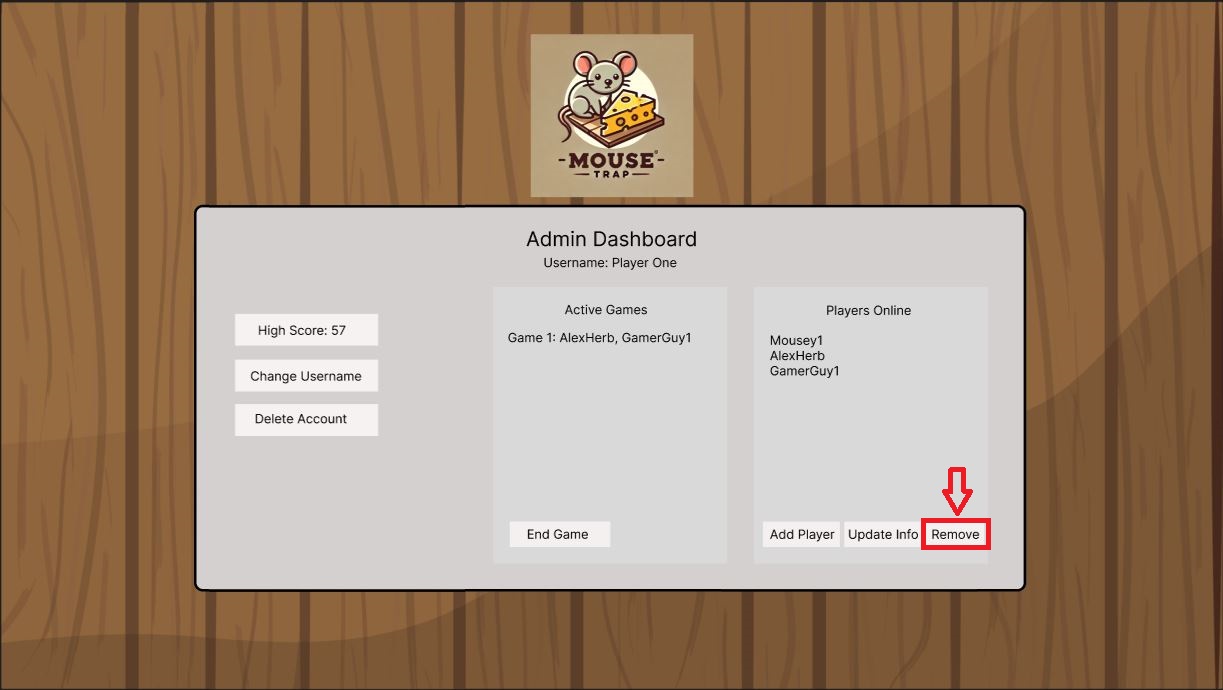
The Administrator account adds a new player by clicking on the “Add Player” button in the player list on the admin dashboard. This then presents a player registration form where the account username and password will be entered to create a new player account.

### User Story 13 – Game Administration, Update data of existing players



The Administrator user selects a player from the list of available players, then clicks the “Update Info” button in the players section of the admin dashboard. This will present them with a form containing the players data, this can then be edited and saved to update the players data.

### User Story 14 – Game Administration, Removing an existing player



The Administrator user selects a player from the list of available players, then clicks on the “Remove” button in the players section of the admin dashboard. This will then prompt the Administrator to confirm their action and the player account is deleted from the database.

## TABLES

To implement a game application system that utilizes a database as the core system that contains the logic, functions, and stores player and game data, tables or entities need to be established that serve a specific purpose within the program. These tables will either be used to store Player information such as username, password, health and current game, or they will be used to store specific game logic and values, such as trap locations, item locations and values, and games that have been started or completed.

### User Table

**Description:** Stores the User data such as login credentials/information, score, and position on the game board.

**Attributes**

* **UserID:** Unique Identifier for individual users.
* **Username:** Players chosen username that is displayed whilst in game.
* **Password:** Players password for accessing their account.
* **LoginAttempt:** Number of failed login attempts by the user. Used to lock account if acceded.
* **LockedAccount:** Boolean Value to check if account is locked.
* **IsAdmin:** Boolean Value to check if user account has admin privileges.
* **Score:** Players highest recorded score.
* **Health:** The current health values for the player.

**Relationships**

* **Inventory Table:** The inventory table is referenced to keep track of what items the user has in their inventory during a game.
* **Game Table:** The game table is referenced to assign the user to the active game they are participating in.
* **Chat Table:** The chat table is referenced to assign messages and the username to the in-game chat functionality. This will display the username in the GUI.
* **Tile Table:** References the Tile table to link the players character to the current tile position on the game board.

### Game Table

**Description:** Stores all data related to active or inactive games, each game has a start and end timestamp, as well as the status of the game.

**Relationships**

* **User Table:** The User table is referenced to connect players to a game and the map.
* **Map Table:** The map table is referenced to assign a map to the current game, this also allows the interactions between the player and the gameboard.
* **Chat Table:** References the Chat table to allow for in-game chat to be recorded and displayed for the current game. Users playing the game will have their username displayed in the chat box.

**Attributes**

* **GameID:** Unique Identifier for individual games in progress.
* **Status:** Status of a game (In progress, waiting, completed)
* **StartTime:** Time that a game was started.
* **EndTime:** Time that a game was completed.

### Item Table

**Description:** Stores all data related to Items that are used/placed in the game.

**Relationships**

* **Inventory Table:** The Inventory Table is referenced by the Item table to allow for items to be picked up and used by the player during a game.
* **Tile Table:** The Tile table is referenced to allow for Items to be placed on tiles available on the game board.
* **ItemType Table:** The ItemType table is referenced to assign the type of item and any associated values.

**Attributes**

* **ItemID:** Unique Identifier for individual game items.
* **Description:** Short Description of the Item.
* **Value:** Stores and values associated with the item.

### ItemType

**Description:** Stores the type of items and any associated values, such as a Cheese for adding points to the players score or Crates that are used to stop the player from entering a tile.

**Relationships:**

* **Item Table:** The Item table is referenced to allow for items to be assigned with a type.

**Attributes**

* **ItemTypeID:** Unique Identifier for the Item Type
* **Type:** String to describe the Item Type, used to determine the type of actions taken within game logic.
* **Point:** Points associated with the item such as health points given or Score to be added to player score.

### Map

**Description:** Stores information related to maps in the game, including home tile positions and maximum rows and columns for the map grid.

**Relationships**

* **Game Table:** The game table is referenced to assign a map to an active game and allow player movement and interaction.
* **Tile Table:** The Tile table is referenced to assign specific tiles and tile types to the map.

**Attributes**

* **MapID:** Unique Identifier for each individual map.
* **HomeTileRowP1:** Row value for the home tile starting position for player 1
* **Home TileColumnP1:** Column value for home tile starting position for player 1.
* **HomeTileRowP2:** Row value for the home tile starting position for player 2
* **Home TileColumnP2:** Column value for home tile starting position for player 2.
* **MaxRows:** Maximum rows available for the map grid.
* **MaxColumns:** Maximum Columns available for the map grid.

### Tile

**Description:** Stores information related to specific tiles that are placed on the game map. Allows for items to be assigned to each tile including barriers of the maze.

**Relationships**

* **Item Table:** This is referenced to allow for items to be placed on specific tiles on the game map.
* **TileType Table:** This reference allows for Tiles to be assigned a specific type such as “Home Tile” and “Barrier” etc.
* **Map Table:** The Map Table is referenced to allow for tiles to be placed on the game map.

**Attributes**

* **TileID:** Unique Identifier for each Tile on the game board.
* **PositionY:** Y position of the tile on the game board.
* **PositionX:** X position of the tile on the game board.

### TileType

**Description:** Stores information related to specific tile types such as “Barrier” and “Home Tile”

**Relationships**

* **Tile Table:** The Tile Table is referenced to assign tiles with specific types.

**Attributes**

* **TileTypeID:** Unique Identifier for the type of tile.
* **Description:** Short description of the type of tile.
* **Value:** Stores any values associated with a tile, used to determine if a player can move into that tile or not.

### Inventory

**Description:** Stores information and data related to the players inventory during an active game, items collected during the game and the number of them are stored here.

**Relationships**

* **User Table:** This table is referenced to assign an inventory to individual users during games.
* **Item Table:** This table is referenced to allow for Items to be added to the inventory as well as being usable by the player during a game.

**Attributes**

* **InventoryID:** Unique Identifier for player inventory.
* **ItemCount:** Used to track the number of items a player has collected during a game, will be used to display the number of items on the GUI.

### Chat

**Description:** Stores messages and data related to in-game chat functionality; these messages will be displayed in the GUI during an active game.

**Relationships**

* **User Table:** This is referenced to allow users to send messages and also display their usernames during an active game.
* **Game Table:** This is referenced to associate an active chat history and messages with an active game.

**Attributes**

* **ChatID:** Unique Identifier for each game chat record.
* **Message:** Stores the messages being sent by users in the in-game chat functionality.

## Mouse Trap Entity RelationShip Diagram