

# **Project Report**

## **Amazing Adventures**

**AA**

*DAT602*

*By Travis Byrman*

# Contents

Milestone One.....	3
Game description.....	3
Storyboard .....	5
Design Choices .....	15
Logical Entity Relationship Diagram .....	16
ERD Analysis.....	17
CRUD Diagram.....	18
CRUD Analysis .....	19
SQL Script .....	21
DDL Script.....	21
Test Queries/Data .....	23
Milestone Two .....	24
ACID.....	24
Multiplayer Support.....	25
Test Data .....	25
Logical Entity Relationship Diagram Alterations.....	26
SQL Procedures and Transactions.....	27
Milestone Three.....	29
Game description - Revised .....	29
Storyboard – Finished Game.....	31
Design Choices .....	44
SQL Procedures and Transactions.....	46
ACID - Revised .....	47
References .....	49

# Milestone One

## Game description

### Idea

I plan to create a single player multicharacter game where characters move around the map via tiles, collecting items and competing with other characters. To achieve this, I will be using a MySQL database along with C# to create an intuitive frontend and efficient backend.

### Gameplay

- The aim of the game is to collect as many items as possible by moving around a tiled map. While doing this, characters must cautiously move around the map, avoiding trapped tiles. If a character moves onto a trapped tile the character will die and lose the game.
- Upon entering into the game, the character will start on the home tile (alternative tile if taken by an existing character).
- Each tile has a maximum of one item or trap. Items and traps are spawned randomly onto tiles.
- The overall game will end when all the characters have quit or died.
- Characters can move up, down, left, or right using arrow keystrokes or the buttons provided.
- If a character leaves the game and the game is still open, their progress will be saved, and they can resume.
- If a tile is occupied by a character, another character cannot be on the same tile.
- When a player tries to join a game, they will be asked to configure their character before joining.
- The game has a maximum time duration. If the game has not been completed before the time runs out, the character scores are calculated, the character with the highest score wins the game.
- Certain players can have administrator abilities. These abilities do not give the player a gameplay advantage but give the player the ability to manage/monitor other players.

### Login and Registration

- Upon logging in, if a player tries to log in with a username that is not established in the database, they will be given the option to register.
- If the player does have an existing account, they will be given the option to try to login again with the correct credentials.

- If the player attempts to login with a valid username, but the password is incorrect, they will be given 5 attempts before the account will be locked.
- If a player account is locked the player will have to ask an Administrator to unlock the account.

### **Lobby**

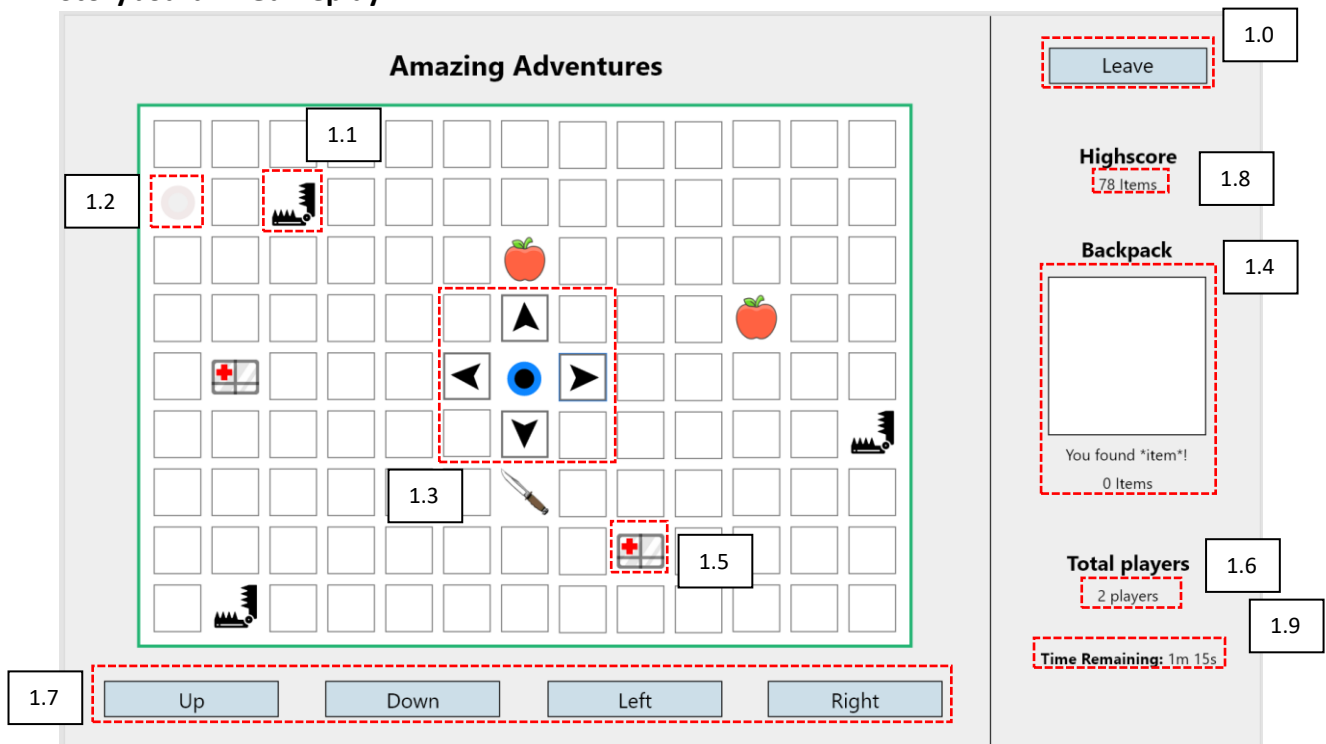
- Inside the lobby, players will be shown the total amount of players and available games they can join.
- Global chat is also available for any player to use.
- Players can also create a game where other players can join.
- If a player is an administrator, they will have access to admin settings.
- Players also have access to their settings, where they can change their username, password etc.

### **Administration Interface**

- After selecting the admin console, administrators will be shown a total list of players and active games.
- From here administrators can lock and change player accounts and delete active games.
- Only certain players will be given administrator abilities.

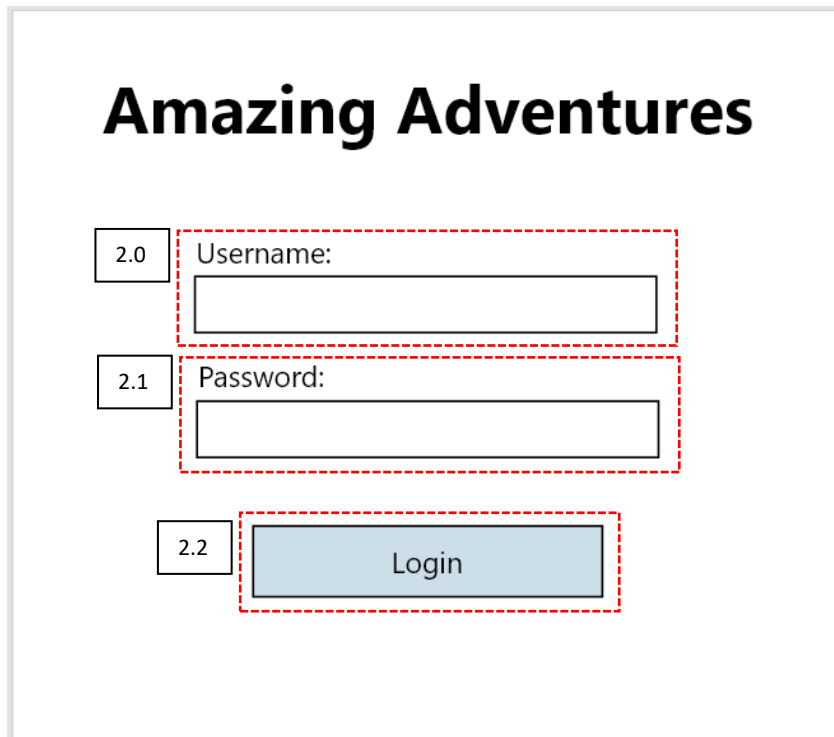
# Storyboard

## Storyboard 1: Gameplay



- 1.0 **Leave button** - Click to exit the game. Player is returned to lobby.
- 1.1 **Trap** – If a character moves to this tile they die and can no longer play.
- 1.2 **Other character** – This is another player.
- 1.3 **Current character** – This is the character that the player is in control of.
- 1.4 **Latest item** – This shows your total items as well as the latest item you have found.
- 1.5 **Item** – This is an item, if a character lands on this tile, they collect the item.
- 1.6 **Total player count** – This shows the total players in the game.
- 1.7 **Character movement** – These buttons can be used to move the character.
- 1.8 **Highscore** – This shows the user their highest score that have received.
- 1.9 **Time remaining** – This shows the player how much time is left.

## Storyboard 2: Login



The storyboard shows a login interface for 'Amazing Adventures'. It features a title at the top, followed by three elements: a username input field (2.0), a password input field (2.1), and a login button (2.2). Each element is highlighted with a red dashed border. The login button is a light blue rectangle with the text 'Login' centered inside.

# Amazing Adventures

2.0 Username:

2.1 Password:

2.2

**2.0 Username field** – Input field where a user inputs their player account username.

**2.1 Password field** – Input field where a user inputs their player account password.

**2.2 Login button** – User clicks this button to login. If username or password is invalid login will be unsuccessful. If username is not a valid player account username, user is given the option to register. User has 5 login attempts before account is locked.

### Storyboard 3: Login – Invalid Attempt

Amazing Adventures

Username:

Password:

Login

3.0 Password Incorrect  
You have 5 attempts remaining

3.0 **Prompt** – Indicates user how many login attempts remaining.

### Storyboard 4: Login – Account locked

Amazing Adventures

Username:

Password:

Login

4.0 Password Incorrect  
This account 'Useraccount123' is **locked**

4.0 **Prompt** – Indicates to user that the account they tried to log into has now been locked.

## Storyboard 5: Login – New Username Detected

Amazing Adventures

Username:

Password:

Login

Register

5.0 New username detected  
Please register above

5.0 **Prompt** – Indicates to user that the username they entered is not recognised in the database. Registration button is shown, given the user the option to register a player with that username.

Amazing Adventures

6.0 Email:

6.1 Username:

6.2 Password:

6.3 Register

6.0 **Email field** – Input field where a user inputs their player account email address.

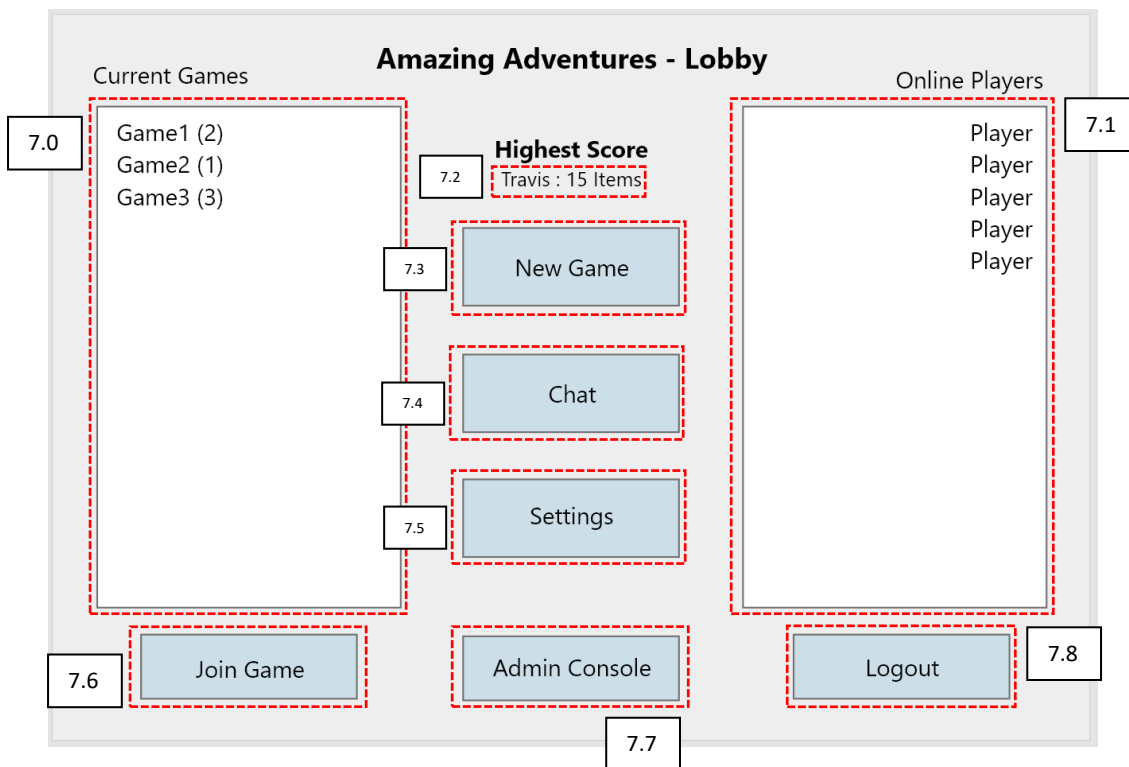
6.1 **Username field** - Input field where a user inputs their player account username.

6.2 **Password field** - Input field where a user inputs their player account password.

6.3 **Register button** – User clicks this button to register the account. If username, password, or email address is invalid, registration will be unsuccessful.



## Storyboard 7: Lobby



7.0 **Current games list** – This shows the current games that players can join.

7.1 **Online players list** – This shows the list of players that are online.

7.2 **High-score** - This shows the players high-score (most items they have collected in a game).

7.3 **New game buttons** – Players can press this button to create a new game.

7.4 **Chat button** – Players can press this button to open the global chat.

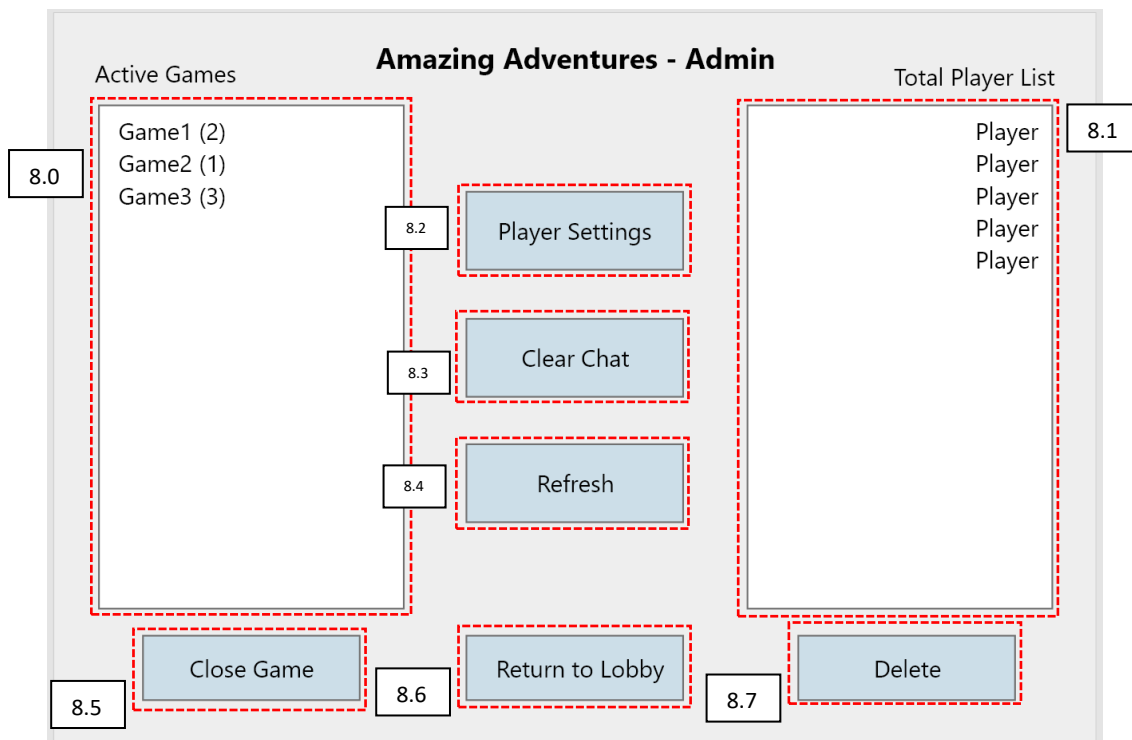
7.5 **Setting buttons** – Players can press this button to view their player settings.

7.6 **Join game button** – Players select a game from the current games list and press this button to join it.

7.7 **Admin console button** – Players can press this button open administrator settings. Only players with administrative access can see this button.

7.8 **Logout button** - Players can press this button logout of their account. User is returned to login screen.

## Storyboard 8: Administrative settings



8.0 **Active games list** – This shows the current games that players can join, as well as the total player count to the left.

8.1 **Total player list** – This shows the list of player accounts created.

8.2 **Player settings button** – Administrators can press this button after selecting a player from the player list to view that players account settings.

8.3 **Clear chat button** – Administrators can press this button to remove all messages from the global chat.

8.4 **Refresh button** – Administrators can press this button to refresh the total player and active games lists.

8.5 **Close game button** – Administrators can press this button after selecting an active game from the list to close/remove the game.

## Storyboard 9: Administrative player settings

**Amazing Adventures - Manage Player**

Username:  9.0

Password:  9.1

Email:  9.2

Highscore:  9.3

Admin: ☐ 9.4

Locked: ☐ 9.5

9.6   9.7

9.0 **Username field** – Input field where a user inputs their player account username.

9.1 **Password field** – Input field where a user inputs their player account password.

9.2 **Email address field** – Input field where a user inputs their player account email address.

9.3 **Highscore field** – Input field where a user inputs their player account highscore.

9.4 **Admin tick box** – Tick box field where a tick represents the user has administrative access.

9.5 **Locked tick box** – Tick box field where a tick represents the players account is locked.

9.6 **Save button** – Administrators click this button to save the players details. If details are incorrectly added, administrator is prompted.

9.7 **Cancel button** – Administrators click this button to cancel the players detail form. This will close the form and return the admin to the administrative settings.

## Storyboard 10: Player settings

The storyboard shows a form titled "Amazing Adventures - Player Settings". It contains three input fields: "Username:", "Password:", and "Email:". Each input field is enclosed in a red dashed border and has a corresponding label box to its right (10.0, 10.1, and 10.2 respectively). Below the input fields are two buttons: "Save" and "Cancel". The "Save" button is enclosed in a red dashed border and has a label box to its left (10.3). The "Cancel" button is also enclosed in a red dashed border and has a label box to its right (10.4).

10.0 **Username field** – Input field where a user inputs their player account username.

10.1 **Password field** – Input field where a user inputs their player account password.

10.2 **Email address field** – Input field where a user inputs their player account email address.

10.3 **Save button** – Players click this button to save their players details. If details are incorrectly added, player is prompted.

10.4 **Cancel button** – Players click this button to cancel their players detail form. This will close the form and return the player to the lobby.

## Storyboard 11: Global chat

The storyboard shows a chat window titled "Amazing Adventures - Chat". Inside the window, there is a text area labeled "Global Chat" containing four lines of test text: "User1: Testetstst", "User2: Testetstst", "User3: Testetstst", and "User1: Testetstst". Below the text area is an input field with a greater-than sign (>) as a placeholder. At the bottom right of the window is a button labeled "Close Chat". Red dashed boxes highlight the text area (11.0), the input field (11.1), and the "Close Chat" button (11.2).

11.0 **Global chat text** – Players will see every player's text chat and date/time of publish.

11.1 **Global chat input box** – Input field where a user inputs the text they wish to add to the chat. Once finished, press keystroke enter to submit.

11.2 **Close chat button** – Players click this button to close the global chat form. This will close the form and return the player to the lobby.

## Storyboard 12: Character creation

The storyboard shows a character creation form titled "Character Select". It contains three input fields: "Character Name" (12.0), "Character Colour" (12.1) with the value "Blue", and two buttons: "Join" (12.2) and "Cancel" (12.3). Red dashed boxes highlight the "Character Name" input field (12.0), the "Character Colour" input field (12.1), the "Join" button (12.2), and the "Cancel" button (12.3).

12.0 **Character name input box** – Input field where a player inputs their character name.

12.1 **Character input colour** – Input field where a player chooses their character colour.

12.2 **Join button** – Players click this button to join the game. This will close the form and will add the player to the game.

12.3 **Cancel button** – Players click this button to close character creation form. This will close the form and return the user to the lobby.

### Storyboard 13: New game form

**New Game**

Name:

Duration:

13.0 **Game name input box** – Input field where a player inputs their game name.

13.1 **Game duration input box** – Input field where a player chooses their games time/duration.

13.2 **Create button** – The player clicks this button to create the game. This will close the form and the player will be asked to create a character.

13.3 **Cancel button** – Players click this button to close the new game form. This will close the form and return the user to the lobby.

### Storyboard 14: Error prompt

**Error!**

An error has occurred:  
**Invalid Input**

Please try again.

13.0 **Ok button** – Players click this button to acknowledge the error.

When an error occurs, this form will pop up notifying the user of their error. The message in red will change depending on the error.

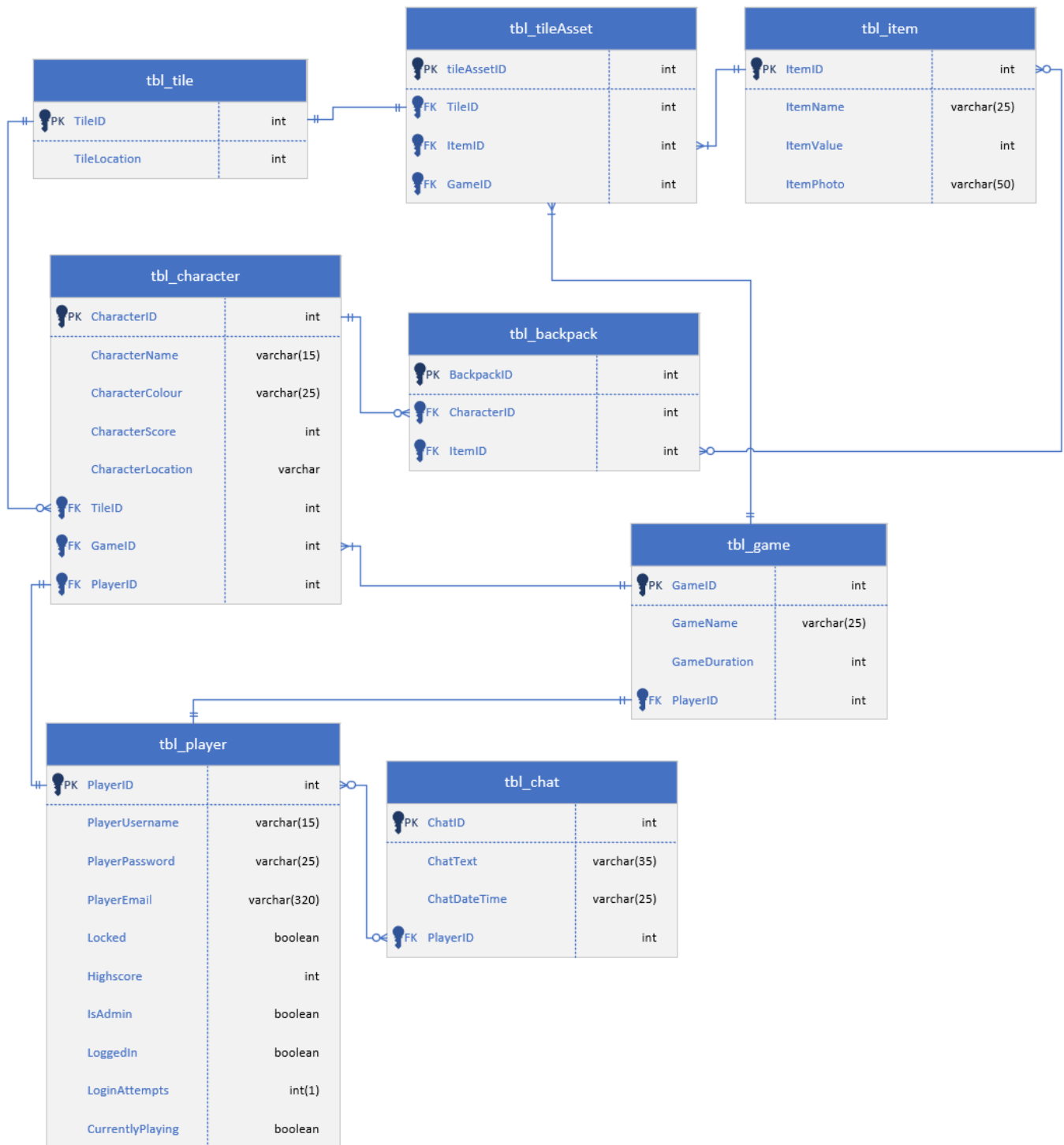
## Design Choices

I chose to have a colour theme that mainly uses the colours grey, white and black. I decided to do this as I found the colours work well with each other and make the features stand out. I found that by using the colour blue and red to show errors and buttons make those features stand out against the grey background. These colours draw the user's attention.

As shown in the gameplay screenshot above, I decided to add buttons that move the player around the map. I chose to do this as it allows users who have a touch screen device to play the game. I also made the text and buttons large enough so that people with disabilities can easily view them. It also allows a large variety of devices to be able to play this game without any difficulties due to resolution.

I have also decided to use combo box when necessary (such as for character colours) as it limits the users range of how many colours they can choose.

## Logical Entity Relationship Diagram





## ERD Analysis

**tbl\_tileID → tbl\_tileAsset**

The tile table contains the location of each tile. Each tile asset needs to be on a tile. One tile can only have one asset, one asset can only be on one tile.

**tbl\_tile → tbl\_character**

The tile table contains the location of each tile. A character needs to be on a tile. One tile can only have one character, one character can only be on a tile.

**tbl\_character → tbl\_backpack**

The character table contains information about the character, such as character name, colour, and score. A character can have multiple items inside the backpack table. A character can have only one item at a move, only one item can be claimed by a character on a tile.

**tbl\_character → tbl\_game**

The character table contains information about the character, such as character name, colour, and score. A character can only be in one game, one game can have many characters.

**tbl\_game → tbl\_tileAsset**

The game table contains information about the game, such as who created the game, game duration and name of the game. One game can have many tile assets, many tile assets can be in one game.

**tbl\_player → tbl\_chat**

The player table contains information about the player, such as username, password, email is it an admin etc. Many players can make submit many texts in the chat table. Many texts can be submitted by many players.

**tbl\_player → tbl\_game**

The player table contains information about the player, such as username, password, email is it an admin etc. Only one player can create a game, the same game can only have one creator (player who creates that exact game).

**tbl\_game → tbl\_character**

The game table contains information about the game, such as who created the game, game duration and name of the game. One game can have many characters, many characters can only be in one game.

**tbl\_item → tbl\_backpack**

The item table contains information about the items, such as item name, item value and item photo. Many items can be in many backpacks, many backpacks can have many items.

**tbl\_item → tbl\_tileAsset**

The item table contains information about the items, such as item name, item value and item photo. The same item can be many tile assets, many tile assets can have the same item.

## CRUD Diagram

Table:	tbl_player	Process:																			
		Register Player	Check Username/Password	Lock Account	Logout Player	Remove Player	Edit Player	Create Character	Start New Game	Join Existing Game	Game Ends	Character Moves	Character Collect's item	Player Adds Chat Message	Player Changes Settings	Admin Creates Player	Admin Removes Player	Admin Locks Player	Admin Edits Player Info	Admin Ends Game	
	PlayerID	C			R	R	R	R	R	R	R			R	R	RC	RD	R	R	R	
	PlayerUsername	C	R				RU							R	C				U		
	PlayerPassword	C	R				RU							R	C				U		
	PlayerEmail	C					RU							R	C				U		
	Locked	C		U													U	U			
	Highscore	C			RU																
	IsAdmin	C				R	R		R						RC	R			RU	R	
	LoggedIn	C	RU		U	R	R									R			R		
	LoginAttempts	C	U																		
	CurrentlyPlaying	C			RU		R	U		U	U								R		
	tbl_character																				
	CharacterID				R			C		C	D	R	R							D	
	CharacterName							C		C	D									D	
	CharacterColour							C		C	D									D	
	CharacterScore				R			U		U	D									D	
	TileID				R			U		U	D									D	
	GameID				R			U		U	D									D	
	PlayerID				R			U		U	D									D	
	tbl_backpack																				
	BackpackID										D		C							D	
	CharacterID										D		U							D	
	ItemID										D		U							D	
	tbl_chat																				
	ChatID													C							
	ChatText													C							
	ChateDateTime													C							
	PlayerID													U							
	tbl_game																				
	GameID								C		D									D	
	GameName								C		D									D	
	Duration								C		D									D	
	PlayerID								U		D									D	
	tbl_item																				
	ItemID																				
	ItemName																				
	ItemValue																				
	ItemPhoto																				
	tbl_tile																				
	TileID																				
	TileLocation																				
	tbl_tileAsset																				
	tileAssetID								C		D		D							D	
	TileID								U		D		D							D	
	ItemID								U		D		D							D	
	GameID								U		D		D							D	

## **CRUD Analysis**

By creating the logical ERD diagram first, I was able to use it as a reference to create the tables and fields for my CRUD diagram. After analysing the assessment requirements and schedule, as well as my storyboards, I would be able to create the processes needed. Here is what I found.

### **Register Player**

When a player enters a username into the login fields, that is not recognised as existing in the database, the user will be prompted. A register button will appear, as well as a label explaining to the user that they can create an account if they do not have an existing account.

After clicking the register button, a registration form will show, getting the user's information to create an account.

### **Check username/password**

When a user submits the login forms, checks will be made with select statements to decide whether the username and password that has been entered is correct. If not correct and the username is recognised, the user will be prompted, as well as the amount of login attempts, they have remaining to access the account.

### **Lock account**

If the user has tried 5 times to log into the same account, the account will become locked. This means that any user will be unable to log into this account without having an administrator unlock it. Once unlocked, users will have another 5 login attempts until the same thing occurs again.

### **Logout player**

If a user logs out of their account, by clicking the logout button in the lobby, the lobby form will close, and the user will be presented with the login form. The account will no longer be shown to be logged in.

### **Remove player**

If an administrator decides to delete another user's player account, they will be asked to confirm this action, then the account will be removed. If the user is currently logged in, they will be logged out immediately. If a user wants to delete their player account they will be prompted to confirm, then they will be logged out.

### **Edit player**

If an administrator wishes to change the settings of another player, they can do so by choosing a player from the player list in the administrator settings. A button will only be visible to administrators on the lobby form. After an admin selects the user, they can press player settings and change the player's settings.

### **Create character**

When a player wants to join a game, they will be prompted to create a character. Here they can choose their character's name and colour before entering. Once entered, the game will be shown, and their character will have the corresponding features.

### **Start new game**

When a player chooses to create a new game, they will be prompted with a form and will need to decide on the games name and duration. If incorrectly submitted, the user will be promoted with the corresponding error message.

### **Join an existing game**

When a player joins an existing game, they will be prompted to create a character (explained above). Then they will have three seconds before the game starts for themselves.

### **Game ends**

When the game ends the player will be given their score. If the score is greater than their highscore, then this new score will be their highscore. Once the game ends, all characters are deleted, and players are returned to the lobby. The game will no longer exist, and players will not be able to join.

### **Character moves**

A player can move their character by using the keystroke arrows or buttons provided on the game play. When moved the database will save the players current location and add any items to their backpack if an item was on the tile.

### **Character collects item**

When a character moves to a tile with an item on it, the item will be removed from the tile and added to their backpack. Depending on the item, the number of points will be determined by it.

### **Player adds chat message**

When a player creates a chat message, it will be shown in the global chat where every player can read it. The players name, chat text and date/time will be provided. Chat messages can be cleared at admins choice in the administration settings.

### **Player changes settings**

If a user wishes to edit their player account details, they can do this by clicking the settings button in the lobby. Here they will be given access to edit the details of their player account. After clicking the save button, the player details will be saved. If incorrectly inputted, the user will be prompted, and the corresponding error message will be shown.

### **Admin creates player**

If an admin wishes to create a player account, they can do so in the administrator settings. After clicking the button, they will be shown a form where they can enter the details. If the details are incorrect the admin will be promoted the account will not be created until errors are corrected.

### **Admin removes player**

If an admin wishes to delete a player than can in the administrator settings by selecting the player and clicking delete.

### **Admin locks player**

If an admin wishes to lock another player account, they can do so in the administrator settings. This will make the account unavailable to any user.

## **Admin ends game**

If an admin wishes to delete/end a game they can do so in the administrator settings by selecting the active game and clicking close game. When they do this, all characters in the game will be deleted and the players will be prompted, that the game ended by an admin. The scores will be shown, and they will be returned to the lobby.

## **SQL Script**

Separate file attached titled AssessmentD DL.sql

## **DDL Script**

### **tbl\_tile**

TileID – Primary key that is an auto-incremental number. This is used to link the tiles to the tileAsset table, and character table.

TileLocation – Integer, used to hold the corresponding value in which the tile can be found in an array.

### **tbl\_tileAsset**

tileAssetID – Primary key that is an auto-incremental number.

This also table consists of three foreign keys, TileID, ItemID and GameID. This table is used identify where and item is, what it is, in what game.

### **tbl\_item**

ItemID - Primary key that is an auto-incremental number. This is used to uniquely identify each item and to link the item to the tileAsset table. This is also used to link the items to the backpack table.

ItemName – Varchar field, used to identify each item by name.

ItemValue – Integer field, used to hold the value of each item.

ItemPhoto – Varchar field, used to hold the link to the item photo in the database.

### **tbl\_character**

CharacterID - Primary key that is an auto-incremental number. This is used to identify each character. This is also used to link the character table to the backpack.

CharacterName – Varchar field, used to identify each character by name.

CharacterColour – Varchar field, used to give each character a colour.

CharacterScore – Int field, used to hold the characters score.

CharacterLocation – Varchar field, used to store the characters current location in the database.

There are also three foreign keys, TileID, GameID and PlayerID. These are used to link the tile, game, and player tables to the character.

### **tbl\_backpack**

BackpackID - Primary key that is an auto-incremental number.

CharacterID – Used to link the character to a backpack that is held by an item.

ItemID – Used to link an item to the backpack that is held by a character.

### **tbl\_player**

PlayerID - Primary key that is an auto-incremental number. This is also used to link the player to the character that they create per game. It is also used to link the player to a game they create. Chat messages from the chat table are also linked to the PlayerID.

PlayerUsername – Varchar field, this is used for users to log into their player account.

PlayerPassword Varchar field, this is used for users to log into their player account securely.

PlayerEmail – Varchar field, this is used for users to associate themselves to their player account securely.

Locked – Boolean, used to identify when the player account is locked or not.

Highscore – Int, used to store the highest score a player gets using their character.

IsAdmin – Boolean, used to identify whether the player has administrator access/powers.

LoggedIn – Boolean, used to identify whether the player is online.

LoginAttempts – Int, used to identify how many times the user has attempted to log into their account.

CurrentlyPlaying – Boolean, used to identify whether the player is in a game or not.

### **tbl\_chat**

ChatID - Primary key that is an auto-incremental number. Used to uniquely identify each chat message.

ChatText – Varchar, used to store the players chat message.

ChatDateTime – Varchar, used to store the date and time that the message was submitted.

PlayerID – Used to link the chat message to the player. Makes the chat message identifiable and traceable back to the player.

### **tbl\_game**

GameID - Primary key that is an auto-incremental number. Used to link the game table with the tileAsset and character table.

GameName – Varchar, used to identify each game by the name given by the user who created the game.

GameDuration – Int, used to store the duration of the game, how long it needs to run for, and how long the duration is left.

PlayerID – Used to identify which player created the game.

## **Test Queries/Data**

As shown in my SQL script, I used a variety of select, update, delete and insert statements to test the functionality of my database.

I used 3 insert statements per table, to give myself enough data to experiment with.

On top of this I used at least one of each select, update, and delete statements for each table. This gave me enough data and statements to work with to test the functionality. From this I found that I needed to add cascade delete statements to my foreign keys to effectively delete records. After adding cascade delete statements, I found the functionality of my database to be great.

# Milestone Two

## ACID

In databases, ACID is an acronym that refers to a set of key properties that affect transactions within a database. A database transaction is 'work' that is performed within a database. A transaction represents the changes and process made to perform the work. If the ACID properties are followed successfully and correctly, then the reliability of the database can be guaranteed.

ACID is broken down into 4 key parts, Atomicity, Consistency, Isolation and Durability.

### Atomicity

Atomicity is when changes to data are performed in a single operation. Atomicity makes sure that the operation is either performed or not, and that the operation is not partially completed. Atomicity is when the operation either it all occurs, or nothing occurs. An example of atomicity is when withdrawing money. An atomic transaction ensures that money is either successfully withdrawn or not withdrawn, never partially withdrawn. For my system atomicity is important to ensure that the game is fair for all players. This means to either complete the operation required or to not complete it. For example, atomicity ensures that the system will complete the games scoring operation accurately. When players receive an item, an operation will occur to reward the player with a point. Atomicity will either increase the players score at a given amount or not, there will never be a 'partial' or half completed operation.

### Consistency

When a consistent transaction begins, the transaction performs an operation at a consistent progress, from beginning to end. The data is in a consistent state until completed. Consistency ensures that the process of data is accurate and consistent for its purpose. An example of a consistent transaction is when money is being transferred. The total balance of the sending and receiving accounts consistently show the exact total after each transfer. For my system consistency is important to ensure that all players in the game are treated the same as other players. Consistency will ensure that every player will be provided with information that is consistently updating, never being outdated. An example of this is when picking up items. Players will be able to see the items they pick up, consistently after they pick up that item.

### Isolation

When an isolated transaction occurs, the transaction will perform independent of other transactions. Isolated transactions do not rely on other transactions, and its success is not influenced by other transactions. An example of this is when money is being transferred between two bank accounts. For money to be transferred, two transactions will take place. The receiving account will perform a transaction that receives the money, and the sending account will perform a transaction that sends the money. Neither transactions will ever interfere with each other or will communicate with each other when isolated. Both bank accounts cannot see the other bank accounts money total. For my system isolation is important to ensure that players do not have access to personal information that only the player itself can know. For example, both players will not be able to see what items they have collected and only will be able to see the total score of each other.



## **Durability**

When a durable transaction occurs, the transaction will be durable and not cause data to fail or go missing. Durable transactions also make sure that data cannot be reverted or changed during the transaction. An example of this is when money is transferred between two bank accounts. When money is withdrawn from one account and deposited into another, the bank account that has the money withdrawn from cannot reverse the transaction, regardless of any factors such as system error. Once a durable transaction begins it cannot be reversed. For my system durability is important to ensure that players cannot reverse the transaction. For example, if a player was locked out of their account due to multiple login attempts, a durable transaction will make sure that the player cannot reverse this operation.

Overall, the properties of ACID are key properties that are very important for data transactions. When taken into consideration, they are very important and useful for data transactions, to ensure that there is no miscommunication or errors occurring during the transaction.

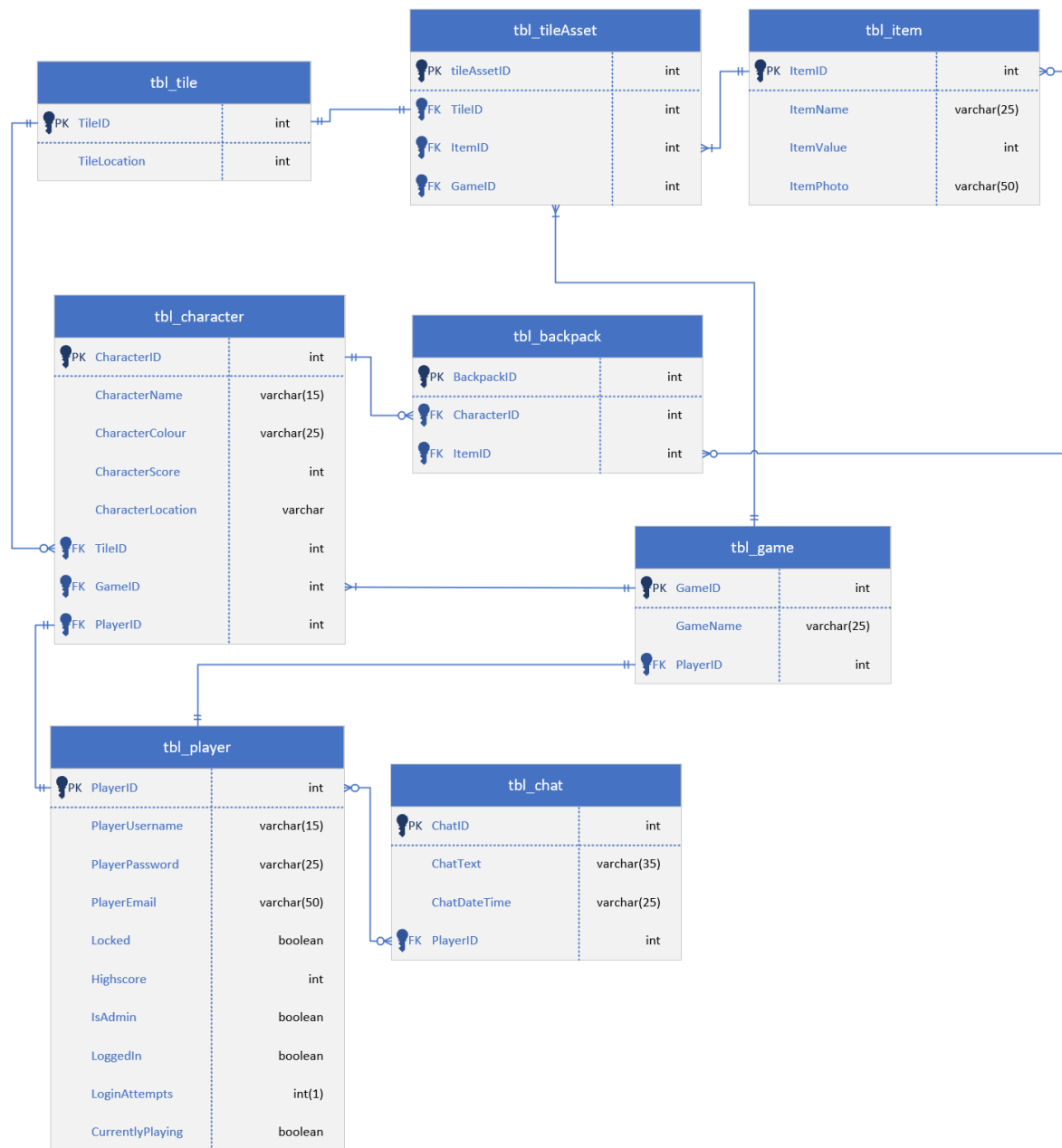
## **Multiplayer Support**

By default, my game should support multiple users as MySQL supports ACID transactions. To ensure that my game is multiplayer compliant, I have created the procedures, keeping the idea in mind that multiplayer support is required. While considering this when I created my procedures, I was able to make sure that players and characters would not get confused with each other, causing multiple records to be affected when a change is made. For example, when a character is to be deleted after the game ends, my 'closeGame' statement will remove all characters that are in the game, by specifying player ID's and game ID's. This ensures that other characters that players have created in other games will not be deleted. I have also implemented tile checking features that take into consideration of other character movements, so that miscalculations and errors will not occur due to confusion.

## **Test Data**

Attached in my MySQL file is test data that can be called using the 'testData' procedure. This is test data that has been improved upon from the first milestone to better suit the changes with my tables. Although test data is supplied, it does not need to be ran as there are two other procedures called 'createDB' and 'modifyDB' which are insert statements to create the tables and records required.

## Logical Entity Relationship Diagram Alterations



After completing my second milestone of my project I decided to make one change to my tables. I decided to remove the game duration field from my game table as I found that I don't need it as a project requirement. Although I could implement this feature, I also realised that it would also make the gameplay less enjoyable as the main focus of the game is to win by having the most points, making a time constraint more stressful and unenjoyable for players.

## SQL Procedures and Transactions

For this milestone I created a series of SQL procedures and transactions that can be split off into five sections. Attached is the SQL file.

### Live game play

Move player/check location – The 'checkCharacterLocation' procedure checks the tile that the character wants to move to. The procedure checks this tile to see if an item or player is on the table. If an item is on the tile, it will be added to the characters backpack. If an existing character is already on this tile, the character will not be able to move to it. If the character lands on a trap, they lose the game, and have to rejoin with a new character.

View backpack – The 'playerBackpack' procedure uses an inner join statement to get the items that are in their backpack, the quantity of the items (if there is more than one of the same item type) and the characters score.

Get the characters location – The 'characterCurrentTile' procedure uses a select statement to get the tile that the character is currently on.

Character leaves game – The 'characterQuits' procedure allows the player to quit the game, with their progress saved. If the player wishes to re-enter the game, they can use the same character and continue with their score until the game ends. This procedure saves the users progress including their tile position.

Character rejoins game – The 'characterRejoin' procedure adds the player back into the game using their existing character. Upon re-joining they start on the same tile they quit the game on, unless a player has taken it.

Adding items to a game – The 'addItem' procedure adds tile assets (items) to the game. The procedure does this by assigning each item with a random tile number. Only one item can occupy a tile, so to make sure this happens each item will be assigned a random tile, until it is confirmed that the tile only occupies one item. The game is limited to have 3 items on the board.

### Player registration

Checking a username – The 'checkUsername' procedure checks the player database to see if the proposed username is taken. If so, the user will have to use a different username that isn't taken.

Creating an account – The 'accountCreate' procedure creates an account using the supplied username, password and email address. This process is done after the 'checkUsername' procedure is called. Administrators also have access to this procedure to create accounts in the interface.

Logging in – The 'accountLogin' procedures checks the given username and password against the player database to make sure that the login details are correct. If the user gets the password wrong for the same username 5 times, then the account is locked and can only be unlocked by an administrator. When the user successfully logs in then the login attempts are 0.

### Player selection

List of games – The 'listOfGames' procedure uses a select statement to provide the player with a list of games available and their corresponding game ID number. Players can then use this ID number to join their desired game.

View game leaderboard – The 'viewLeaderboard' procedure uses a select statement to provide the player with the leaderboard of their chosen game. This procedure shows the game name, players that have joined the game and their character score.

Character creation/joining – The 'characterJoinGame' procedure creates a character, personalised by the player. Each game that a player joins, they are required to create a new character (unless returning to a game they have already joined).

Add chat text – The 'globalChat' procedure creates chat messages and allows players to view the chat. Each chat message has the players username, date/time and chat message.

### **Confirmation for a game**

Creating a game – The 'gameCreation' procedure creates a game using the players username and game name decided by the player. If the player has already created a game, they will not be able to create another game until they close the game.

Close your own game – The 'gameClose' procedure allows players to close the game that they created. This will delete all characters, remove the game, and set the players highscore as their character scores if their character scores are higher than their highscore.

### **Game administration functions**

Change email – The 'changeEmail' procedure allows players to change their email address. It also allows administrators to change other players email addresses. The same email address can be used for multiple accounts.

Change password – The 'changePassword' procedure allows players to change their password. It also allows administrators to change other players account password.

Lock player – The 'lockPlayer' procedure allows administrators to lock player accounts, banning them from being used. Administrators can unlock locked accounts the same way.

Delete player – The 'deletePlayer' procedure allows administrators to delete player accounts. This allows other users to create accounts with deleted player usernames.

Assign admin permissions – The 'adminSetAdmin' procedure allows administrators to give other players administrator access. Administrators can also take administrator access away from players.

Clear global chat – The 'clearGlobalChat' procedure allows administrators to remove all chat messages from the global chat.

Close a game – The 'adminCloseGame' procedure allows administrators to close other players games. This will delete all characters, remove the game, and set the players highscore as their character scores if their character scores are higher than their highscore.

# Milestone Three

## Game description - Revised

After analysing and reading the assessment criteria more carefully, I realised that my gameplay is overcomplicated and, in some parts, not following the assessment criteria. Because of this I changed how the game is played. Highlighted in bold are the changes I made.

### Gameplay

- The aim of the game is to collect as many items as possible by moving around a tiled map. While doing this, characters must cautiously move around the map, avoiding trapped tiles. If a character moves onto a trapped tile the character will die and lose the game.
- Upon entering into the game, the character will start on the home tile (alternative tile if taken by an existing character).
- Each tile has a maximum of one item or trap. Items and traps are spawned randomly onto tiles.
- **The overall game will end when the game creator creates a new game, or an admin ends the game.** – Changed to better suit the gameplay, allowing more game options without having to create a game.
- Characters can move up, down, left, or right using arrow keystrokes or the on-screen buttons provided.
- If a character leaves the game and the game is still open, their progress will be saved, and they can resume when the character re-joins.
- If a tile is occupied by a character, another character cannot move to the occupied tile.
- When a player tries to join a game, they will be asked to configure their character before joining.
- **No time duration** – Time duration removed to better suit the gameplay. The game is more enjoyable when not constrained by time. Was not needed for the assessment criteria.
- Certain players can have administrator abilities. These abilities do not give the player a gameplay advantage but give the player the ability to manage/monitor other players.
- **Traps that are on tiles are in visible. Players cannot see where the traps are located** – Added to make the gameplay more exciting and competitive for the players. This adds an element of difficulty to the game.

## Login and Registration

- If a player tries to log in with a username that is not recognised by the database, **a password input field will be shown, and they will be prompted to enter a password to create an account with the entered username.** – Removed registration form to better suit/follow the assessment criteria.
- If the player enters a username **recognised by the database, a password input field will be shown. The player will be prompted that an account is recognised, and to enter the accounts password.** – Added to better suit/follow the assessment criteria.
- If the player attempts to login with a valid username, but the password is incorrect, they will be given 5 attempts before the account will be locked.
- If a player account is locked the player will have to ask an Administrator to unlock the account and will not be able to log into that account until it is unlocked.

## Lobby

- Inside the lobby, players will be shown the total amount of online players and available games they can join. **The online player list will also contain the players highest score achieved.**
- Global chat is also available for any player to use.
- Players can create a game where other players can join.
- **Players can click a refresh button to refresh the game and player lists.**
- If a player is an administrator, they will have access to administrator settings.
- Players also have access to their settings, where they can change **their password and add an email address.** – Users cannot change their username, only admins. This is to prevent conflicting/inappropriate usernames.

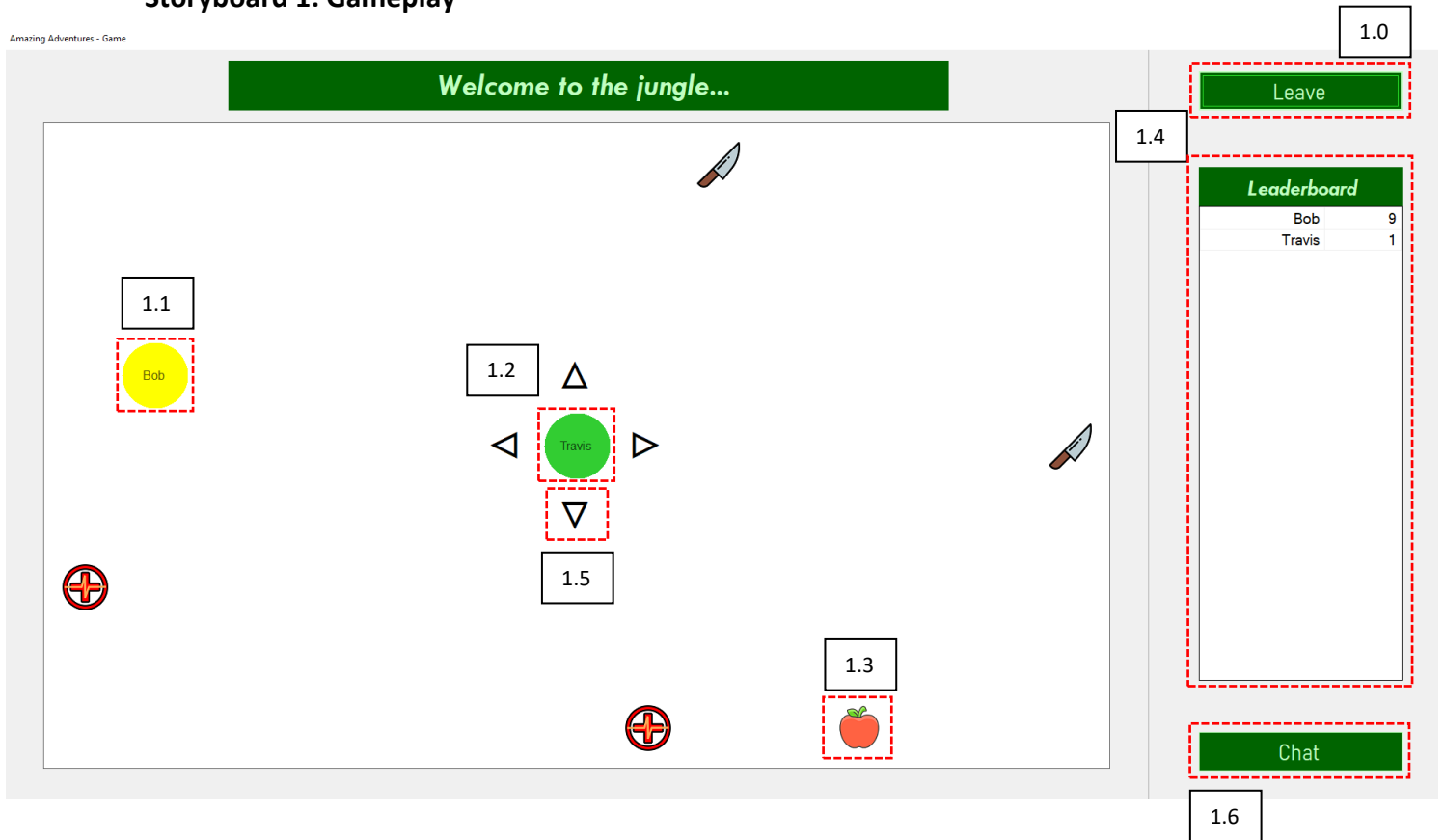
## Administration Interface

- After entering the admin settings, administrators will be shown the overall list of players and active games.
- **Administrators can then change player settings.**
- **Administrators can then delete games and players.**
- **Administrators can clear the global chat.**
- **Administrators can create player accounts.**
- Only certain players will be given administrator abilities.

# Storyboard – Finished Game

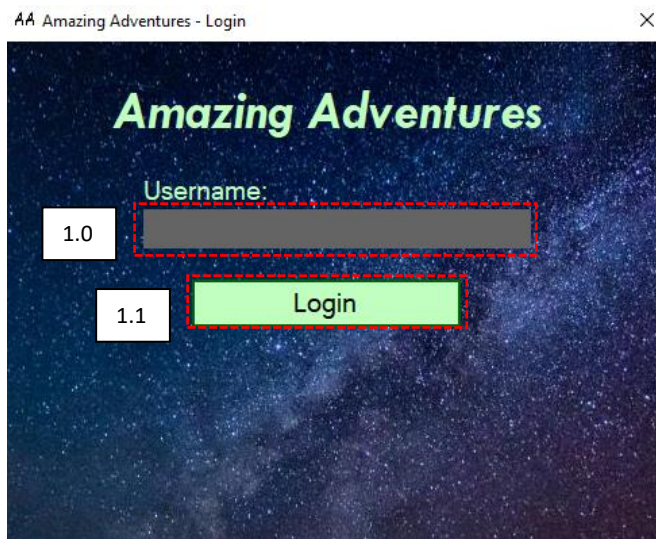
## Storyboard 1: Gameplay

Amazing Adventures - Game



- 1.0 **Leave button** - Click to exit the game. Player is returned to lobby.
- 1.1 **Other character** – This is another player.
- 1.2 **Current character** – This is the character that the player is in control of.
- 1.3 **Item** – This is an item, if a character lands on this tile, they collect the item.
- 1.4 **Leaderboard** – This shows the characters playing in the game and their score.
- 1.5 **Character movement** – These buttons can be used to move the character.
- 1.6 **Chat button** – Click to show the global chat.

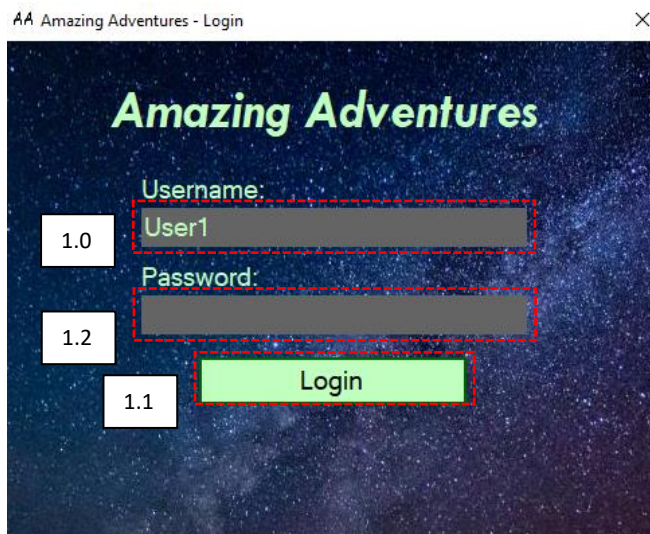
## Storyboard 2: Login



1.0 **Username input** – Input used to enter username.

1.1 **Login button** – Click this button to check username.

## Storyboard 3: Login – Recognised username



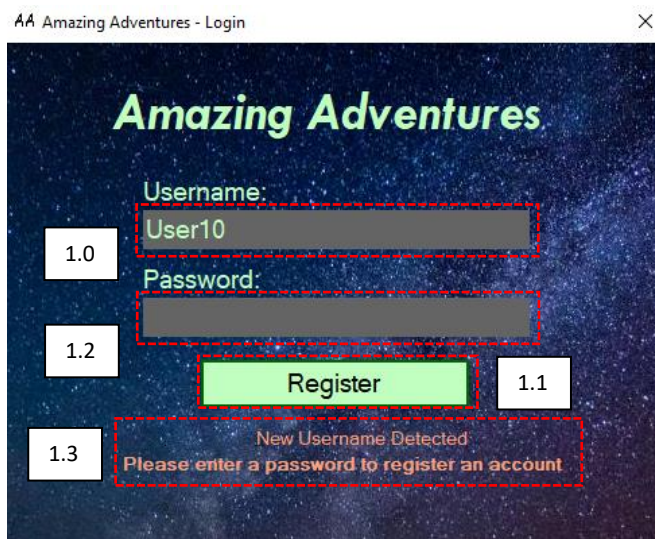
1.0 **Username input** – Input used to enter username.

1.1 **Login button** – Click this button to check username.

1.2 **Password Input** – Input used to enter password.

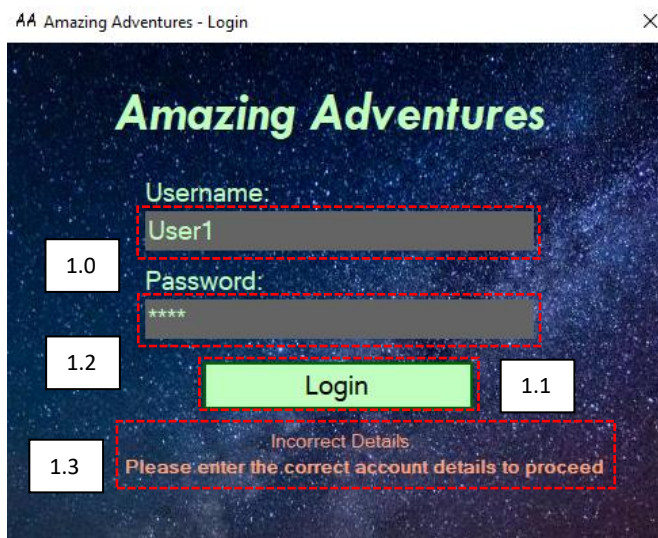


### Storyboard 3: Login – Unrecognised username



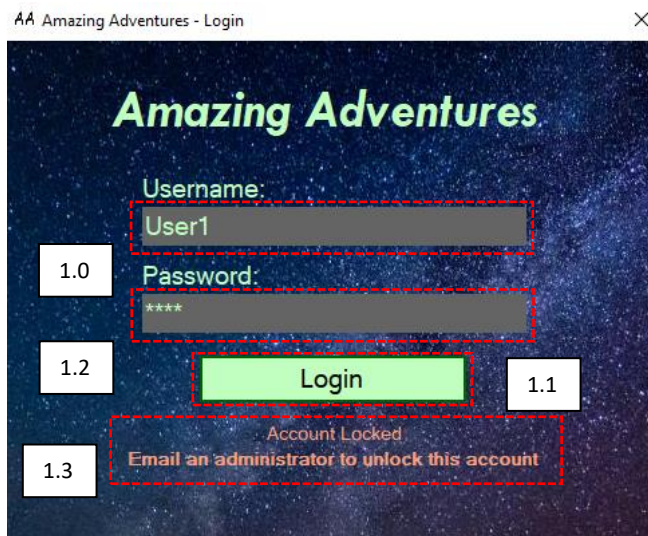
- 1.0 **Username input** – Input used to enter username.
- 1.1 **Login button** – Click this button to check username.
- 1.2 **Password Input** – Input used to enter password.
- 1.3 **Login Error Message** – To show user that the username is not linked to an account.

### Storyboard 4: Login – Incorrect login credentials



- 1.0 **Username input** – Input used to enter username.
- 1.1 **Login button** – Click this button to check username.
- 1.2 **Password Input** – Input used to enter password.
- 1.3 **Login Error Message** – To show user that the login credentials are incorrect.

## Storyboard 5: Login – Account locked



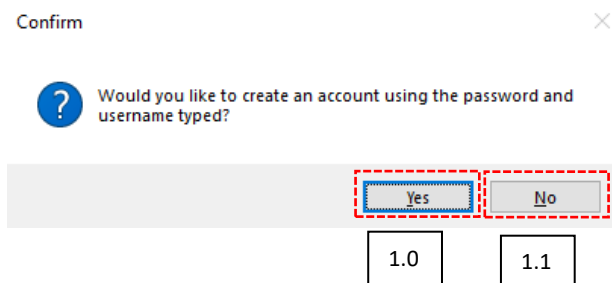
1.0 **Username input** – Input used to enter username.

1.1 **Login button** – Click this button to check username.

1.2 **Password Input** – Input used to enter password.

1.3 **Login Error Message** – To show user that the account is locked.

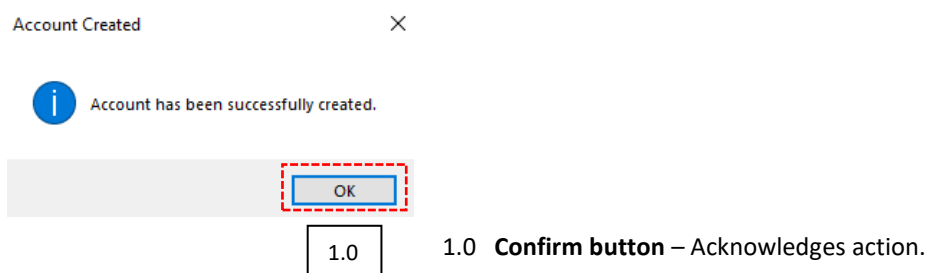
## Storyboard 6: Login – Account register prompt



1.0 **Confirm button** – Creates user account and logs user in.

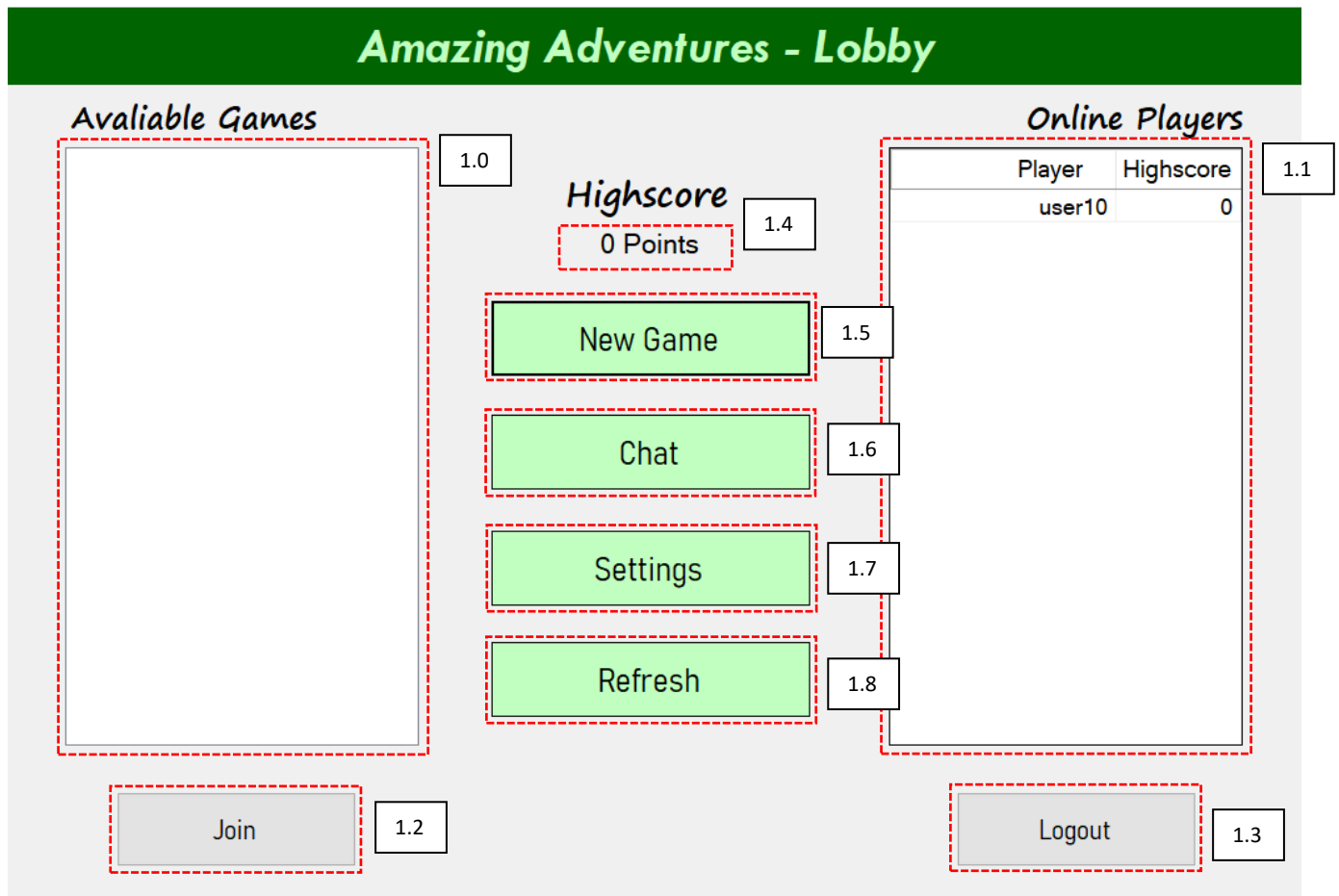
1.1 **Cancel button** – Does not create user account.

## Storyboard 7: Login – Account register confirmation



## Storyboard 8: Lobby

Amazing Adventures - Lobby



- 1.0 **Available games list** – Shows list of games open. If empty, no games are open.
- 1.1 **Online player list** – Shows list of players online and their highscores.
- 1.2 **Join game button** – Click to join selected game from games list.
- 1.3 **Logout** – Click to log out of the game.
- 1.4 **Highscore** – Shows the player that is logged in their highest score.
- 1.5 **New game button** – Click to create a new game.
- 1.6 **Chat button** – Click to view the global chat.
- 1.7 **Settings button** – Click to view the players settings.
- 1.8 **Refresh button** – Click to refresh the game and player lists.

## Storyboard 9: Create game

Amazing Adeventures - Create Game

The storyboard for 'Create game' consists of three frames. Frame 1.0 shows a green header bar with the text 'Create Game' in white. Below the header is a text input field with a red dashed border, preceded by the label 'Name:'. Frame 1.1 shows a grey 'Cancel' button with a red dashed border. Frame 1.2 shows a blue 'Create' button with a red dashed border.

1.0 **Game name input** – Input used to enter game name.

1.1 **Cancel button** – Click to cancel the game creation.

1.2 **Create button** – Click to create the game.

## Storyboard 10: Create character

Amazing Adventures - Create Character

The storyboard for 'Create character' consists of four frames. Frame 1.0 shows a green header bar with the text 'Create Character' in white. Below the header is a text input field with a red dashed border, preceded by the label 'Character name:'. Frame 1.1 shows a 'Pick Colour' button with a red dashed border, preceded by the label 'Character colour:'. Frame 1.2 shows a grey 'Cancel' button with a red dashed border. Frame 1.3 shows a blue 'Create' button with a red dashed border.

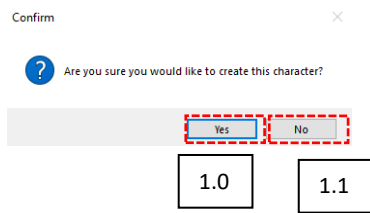
1.0 **Character name input** – Input used to enter character name.

1.1 **Character colour picker** – Click to pick character colour.

1.2 **Cancel character cancel button** – Click to cancel the character creation.

1.3 **Create character cancel button** – Click to create the character.

## Storyboard 11: Create character prompt

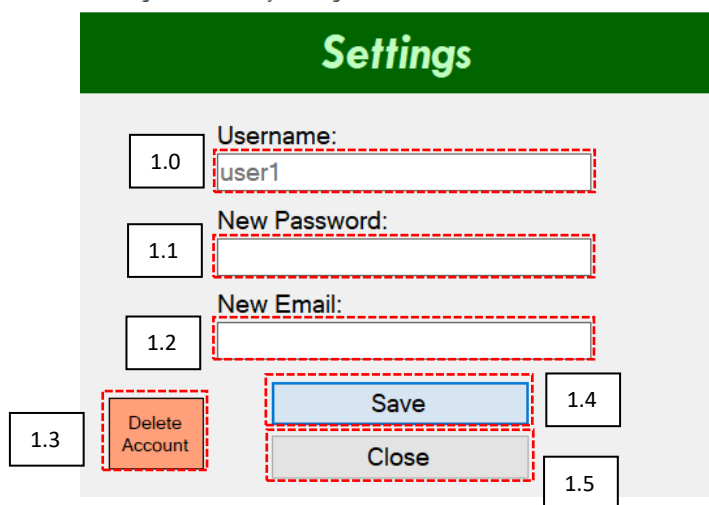


1.0 **Confirm button** – Create character and joins game.

1.1 **Cancel button** – Does not create the character.

## Storyboard 12: Player settings

Amazing Adventures - Player Settings



1.0 **Username input** – Input used to enter username.

1.1 **Password input** – Input used to enter password.

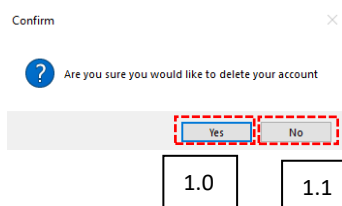
1.2 **Email input** – Input used to enter email address.

1.3 **Delete account button** – Click to delete the player's account.

1.4 **Save settings button** – Click to save the players new settings.

1.5 **Close settings button** – Click to close the players settings.

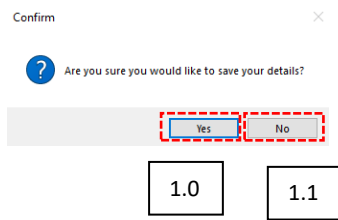
## Storyboard 13: Delete player account



1.0 **Confirm button** – Deletes user account and signs user out.

1.1 **Cancel button** – Does not delete the user account.

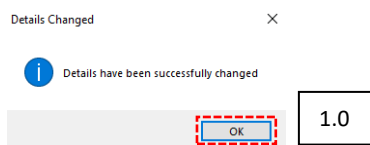
## Storyboard 14: Save player details prompt



1.0 **Confirm button** – Deletes user account and signs user out.

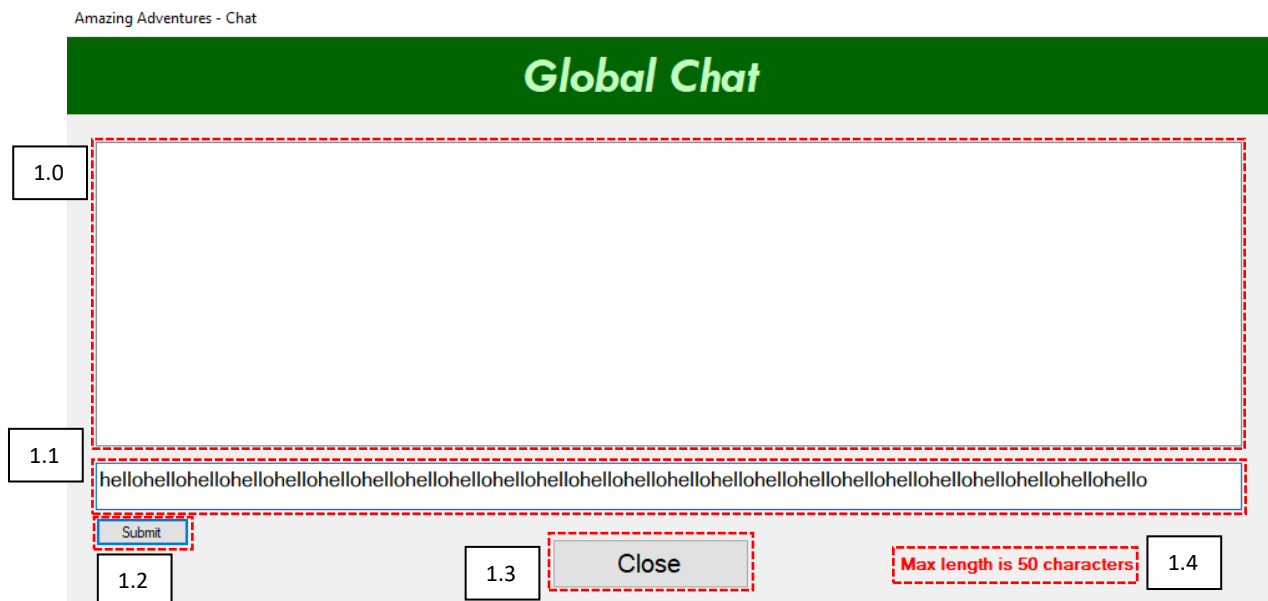
1.1 **Cancel button** – Does not delete the user account.

## Storyboard 15: Details changed prompt



1.0 **Ok button** – Click to acknowledge change.

## Storyboard 16: Global chat



1.0 **Chat messages** – Shows all user messages, including date, time, username, and message.

1.1 **Chat input field** – Does not delete the user account.

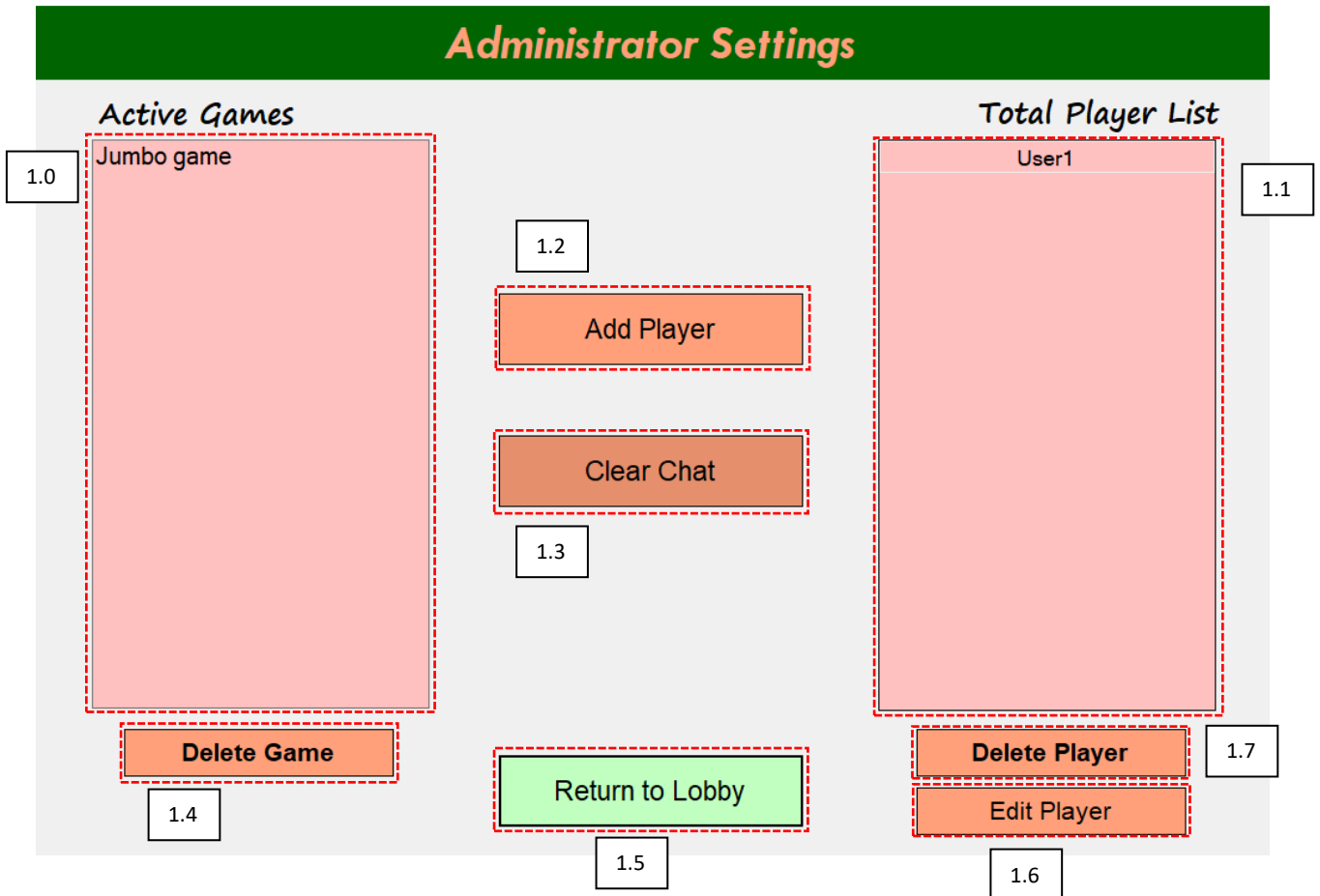
1.2 **Chat message send button** – Does not delete the user account.

1.3 **Chat close button** – Click to close global chat.

1.4 **Max character message** – This message shows when a player creates a message that's more than 50 characters.

## Storyboard 17: Administrator settings

Amazing Adventures - Admin Settings



- 1.0 **Active games list** – Shows list of all active games.
- 1.1 **Total player list** – Shows list of total player accounts created.
- 1.2 **Add player button** – Click to create a player account.
- 1.3 **Clear chat button** – Click to remove all chat messages from global chat.
- 1.4 **Delete game button** – Click to delete a game selected from the games list.
- 1.5 **Return to lobby button** – Click to return to the lobby menu.
- 1.6 **Edit player button** – Click to edit player selected from the player list.
- 1.7 **Delete player button** – Click to delete player selected from the player list.

## Storyboard 18: Administrator - Create player

Amazing Adventures - Create Player

The image shows a 'Create Player' form with a green header. The form contains four input fields: a 'Username' field (1.0), a 'Password' field (1.1), a 'Create' button (1.2), and a 'Close' button (1.3). The 'Create' button is blue and the 'Close' button is grey. Both buttons are highlighted with red dashed boxes. The 'Close' button is positioned below the 'Create' button. The 'Create' button is positioned to the right of the 'Password' field. The 'Close' button is positioned to the right of the 'Create' button. The 'Create' button is positioned to the right of the 'Password' field. The 'Close' button is positioned to the right of the 'Create' button.

- 1.0 **Username input field** – Used to enter new player account username.
- 1.1 **Password input field** – Used to enter new player account password.
- 1.2 **Create account button** – Click to create a player account.
- 1.3 **Close form button** – Click to close create player form.

## Storyboard 19: Administrator - Confirm player create prompt

The image shows a 'Confirm' dialog box with a question mark icon and the text 'Are you sure you would like to create this player?'. Below the question are two buttons: 'Yes' (1.0) and 'No' (1.1). Both buttons are highlighted with red dashed boxes. The 'Yes' button is positioned to the left of the 'No' button. The 'Yes' button is positioned to the left of the 'No' button.

- 1.0 **Confirm player create button** – Used to confirm the creation of a new player.
- 1.1 **Cancel player create button** – Used to cancel the creation of a new player.

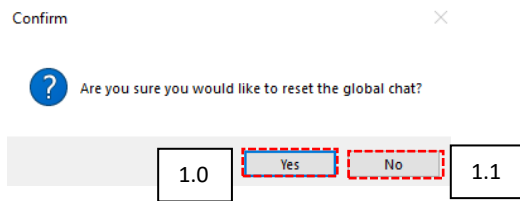
## Storyboard 20: Administrator - Player create confirmation

The image shows an 'Account Created' dialog box with an information icon and the text 'Player account is created'. Below the text is a single button labeled 'OK' (1.0). The 'OK' button is highlighted with a red dashed box. The 'OK' button is positioned to the left of the '1.0' label.

- 1.0 **Ok button** – Click to acknowledge player creation.



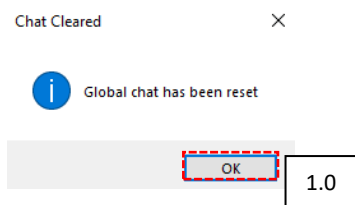
## Storyboard 21: Administrator - Clear chat prompt



1.0 **Confirm clear chat button** – Click to remove all messages from the global chat.

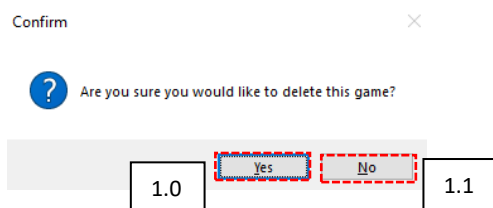
1.1 **Cancel clear chat button** – Click to cancel the removal of all messages from the global chat.

## Storyboard 22: Administrator - Clear chat confirmation



1.0 **Ok button** – Click to acknowledge the global chat has been cleared.

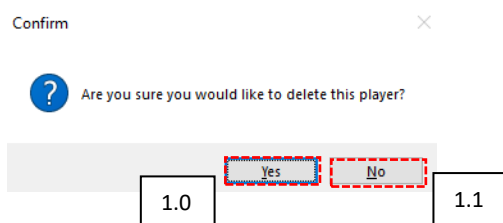
## Storyboard 23: Administrator - Delete game prompt



1.0 **Confirm game delete button** – Click to delete selected game from games list.

1.1 **Cancel game delete button** – Click to cancel the deletion of game from games list.

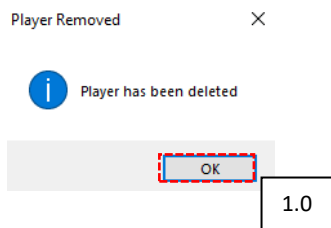
## Storyboard 24: Administrator - Delete player prompt



1.0 **Confirm player delete button** – Click to delete selected player from players list.

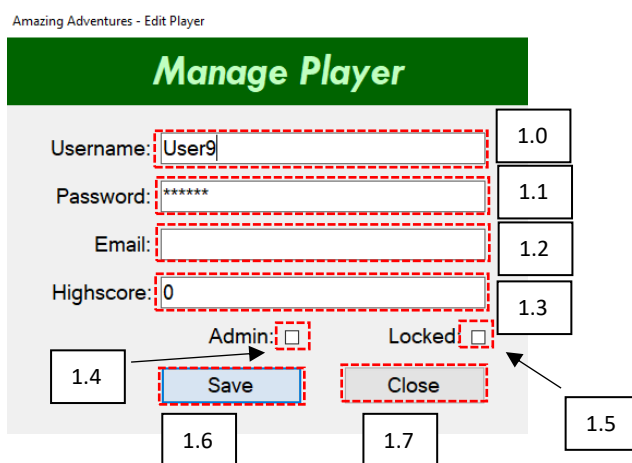
1.1 **Cancel player delete button** – Click to cancel the deletion of player from players list.

## Storyboard 25: Administrator - Delete player confirmation



1.0 **Ok button** – Click to acknowledge the player delete.

## Storyboard 26: Administrator - Manage player



1.0 **Username input field** – Used to enter player account username.

1.1 **Password input field** – Used to enter player account password.

1.2 **Email address input field** – Used to enter player account email address.

1.3 **Highscore input field** – Used to enter player highscore.

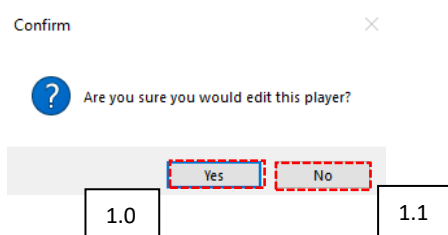
1.4 **Admin checkbox** – Tick to make user an administrator.

1.5 **Locked checkbox** – Tick to lock user account.

1.6 **Save form button** – Click to save player details.

1.7 **Close form button** – Click to close player details form.

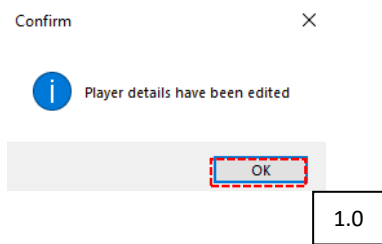
## Storyboard 27: Administrator - Manage player prompt



1.0 **Confirm edit player button** – Click to confirm edit selected player details from players list.

1.1 **Cancel edit player button** – Click to cancel edit of player details from players list.

## Storyboard 28: Administrator - Manage player confirmation



1.0 **Ok button** – Click to acknowledge the player edit.

## Design Choices

For the final GUI I decided to follow the design choices I made in Milestone One. Although will creating the GUI I felt that the use of just white, black, and grey is too boring. These colours are also typical normal colours, that make elements feel 'normal', nothing special. Because of this I used a few other colours such as variations of green and blue to make elements stand out. I also used a background image for a few forms, to show the idea of there being an adventure. One of the wallpapers was of stars, to show that the sky is the limit for an adventure.

As shown in my storyboards above, I also used the colour red. I used this colour to show the user that the element is important. Typically, red is represented to mean 'stop', which makes the user psychologically think not to press it immediately without some thought. I used red on important elements, such as deleting players, editing players, deleting games etc, elements that would impact the game largely.

Shown below is the colour palette I used.

	#FFFFFF	Form/list/data grid view colour
	#000000	Text colour
	#F0F0F0	Form colour
	#006400	Text background colour
	#C0FFC0	Button colour
	#FFC0C0	List box/data grid view colour
	#FFA07A	Button colour
	#646464	Input/panel colour
	#D7F4E2	Text colour
	#0078D7	Button border colour
	#32CD32	Default character colour

I also did not implement certain elements that I described in the milestone one storyboard.

**Register form** – I did not have a register account form as described in the original storyboards. This was because after I re-read the criteria the login functionality was not meant to be designed like that. The criteria states that it wants users to signup repurposing the login form depending on the scenario. Because of this I removed my registration form and followed those outlines. This actually made my GUI better as less forms were used, making it less complicated for the user. Because of this I also made the email address field not required and can be left blank. Although users can still add an email address to their account manually.

I also added an extra element that I didn't have on my milestone one storyboard.

**Account delete button** – I added an account delete button in the player settings form. This is because I re-read the criteria and noticed that I am required to implement a way for players to delete their own account. I originally did not have this element on my storyboard.

I also designed some elements of the GUI to not function a particular way. If an admin wishes to edit their player account details, like how they would edit other players details, they cannot. I have disabled this so that players cannot change the properties of themselves, e.g username, admin privileges etc. If a player does do this, then the changes may negatively impact the game. It also prevents users from hacking/accessing an administrator's account without permission and doing something to the account. This also stops administrators from altering there own highscore.

## SQL Procedures and Transactions

For this milestone I made a few changes to my procedures and transactions. While developing the GUI I noticed that I had some procedures that I did not need and some that I needed but did not have. Below are the changes.

I also added transactions to my procedures, whereas before I did not have any transactions.

I also limited the chat length to 50 characters for each global chat message.

### New Procedures

Check character existence – The 'CheckCharacter' procedure checks to see if a player has already created a character for a game. If so the game will use that character, if not the player will need to create a character.

Get character colour – The 'GetCharacterColor' procedure gets the colour of the character from the game. This is to set the colour of the character marker on the game board.

Get all character positions in game – The 'GetAllCharacterPositions' procedure gets all the positions of the characters in the game so that the character markers are placed on the correct tiles on the game board.

Check admin – The 'CheckAdmin' procedure checks if a player is an admin. If so the admin settings are available to the user, otherwise the user cannot see the admin settings.

Get items from game – The 'GetGameItems' procedure gets all the items from the game. This is so that all the items can be displayed on the correct tiles on the game board.

Get character score from game – The 'GetCharacterScore' procedure gets the score of a specific character and shows it on the game lose form.

Check if game exists – The 'CheckGameExists' procedure checks if a player has already created a game. If so the player is asked to delete the game before they create a new one. Otherwise, the player can create a game.

Get player information – The 'GetPlayerInfo' procedure gets all the players information and populates the player details form.

Get scores of characters from game – The 'GetGameCharacterScores' procedure gets all the scores and names of the characters from a game. These scores and names are then shown on the game leaderboard.

Update player information – The 'UpdatePlayerInfo' procedure updates the players information when the change is actioned on the player details form.

Total list of players – The 'ListOfPlayers' procedure gets all the username of every player account created. This is used in the admin settings for the data grid view.

### Procedures Removed

Player backpack – The 'PlayerBackpack' procedure was removed as I re-read the assessment criteria and realised that I didn't need it. I also found that the backpack table is not really being utilised, but regardless of this, I decided to keep it anyway. This is because the backpack table does work as intended and is a great feature to add for future implementation. I feel that giving players a way to

view their backpack would give the game an additional element of information, although right now it is not an essential feature.

Set admin access – The “AdminSetAdmin” procedure was removed as there was not need for it. If the admin status of a player needs changing, then an admin can do this in the player settings of the particular player.

## **ACID - Revised**

In databases, ACID is an acronym that refers to a set of key properties that affect transactions within a database. A database transaction is ‘work’ that is performed within a database. A transaction represents the changes and process made to perform the work. If the ACID properties are followed successfully and correctly, then the reliability of the database can be guaranteed.

ACID is broken down into 4 key parts, Atomicity, Consistency, Isolation and Durability.

### **Atomicity**

Atomicity is when changes to data are performed in a single operation. Atomicity make sures that the operation is either performed or not, and that the operation is not partially completed. Atomicity is when the operation either it all occurs, or nothing occurs. An example of atomicity is when withdrawing money. An atomic transaction ensures that money is either successfully withdrawn or not withdrawn, never partially withdrawn. For my system atomicity is important to ensure that the game is fair for all players. This means to either complete the operation required or to not complete it. For example, atomicity ensures that the system will complete the games scoring operation accurately. When players receive an item, an operation will occur to reward the player with a point. Atomicity will either increase the players score at a given amount or not, there will never be a ‘partial’ or half completed operation.

### **Consistency**

When a consistent transaction begins, the transaction performs an operation at a consistent progress, from beginning to end. The data is in a consistent state until completed. Consistency ensures that the process of data is accurate and consistent for its purpose. An example of a consistent transaction is when money is being transferred. The total balance of the sending and receiving accounts consistently show the exact total after each transfer. For my system consistency is important to ensure that all players in the game are treated the same as other players. Consistency will ensure that every player will be provided with information that is consistently updating, never being outdated. An example of this is when picking up items. Players will be able to see the items they pick up, consistently after they pick up that item.

### **Isolation**

When an isolated transaction occurs, the transaction will perform independent of other transactions. Isolated transactions do not rely on other transactions, and its success is not influenced by other transactions. An example of this is when money is being transferred between two bank accounts. For money to be transferred, two transactions will take place. The receiving account will perform a transaction that receives the money, and the sending account will perform a transaction that sends the money. Neither transactions will interfere with each other or will communicate with each other when isolated. Both bank accounts cannot see the other bank accounts money total. For my system isolation is important to ensure that players do not have access to personal information

that only the player itself can know. For example, both players will not be able to see what items they have collected and only will be able to see the total score of each other.

### **Durability**

When a durable transaction occurs, the transaction will be durable and not cause data to fail or go missing. Durable transactions also make sure that data cannot be reverted or changed during the transaction. An example of this is when money is transferred between two bank accounts. When money is withdrawn from one account and deposited into another, the bank account that has the money withdrawn from cannot reverse the transaction, regardless of any factors such as system error. Once a durable transaction begins it cannot be reversed. For my system durability is important to ensure that players cannot reverse the transaction. For example, if a player was locked out of their account due to multiple login attempts, a durable transaction will make sure that the player cannot reverse this operation.

Overall, the properties of ACID are key properties that are very important for data transactions. When taken into consideration, they are very important and useful for data transactions, to ensure that there is no miscommunication or errors occurring during the transaction.

For my project MySQL implements ACID via InnoDB transactions. InnoDB is a storage engine that MySQL utilises to process data statements. MySQL uses three statements to achieve atomicity.

**AUTOCOMMIT** – Processes statements automatically. This statement is also used to achieve consistency, as when used, commits all statements at point of execution.

**COMMIT** – Processes all statements when this statement is processed.

**ROLLBACK** – Reverts commit when this statement is processed.

To achieve consistency InnoDB has a crash recovery process. When a MySQL database crashes, InnoDB uses the logs to perform a roll-forward of the database. InnoDB also rolls back any uncommitted transactions. To achieve durability, MySQL has software to assist interactions with the hardware utilised by the MySQL server. These features make the interactions between the hardware and the server efficient and effective. These features put measures in place in the event that a hardware fault effects the data transfers.



## References

*IBM docs.* (n.d.). IBM - United States. Retrieved May 25, 2021,

from <https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions>

*What is atomicity? - Definition from Techopedia.* (2011, September 5). Techopedia.com.

Retrieved May 25, 2021,

from <https://www.techopedia.com/definition/24729/atomicity>

*MySQL :: MySQL 5.6 reference manual :: 14.1 introduction to InnoDB.* (n.d.). MySQL :: Developer

Zone. Retrieved June 27, 2021,

from <https://dev.mysql.com/doc/refman/5.6/en/innodb-introduction.html#:~:text=InnoDB%20is%20a%20general%2Dpurpose,clause%20creates%20an%20InnoDB%20table>

*MySQL :: MySQL 5.6 reference manual :: 14.2 InnoDB and the ACID model.* (n.d.). MySQL ::

Developer Zone. Retrieved June 27, 2021,

from <https://dev.mysql.com/doc/refman/5.6/en/mysql-acid.html>