Project Report

Amazing Adventures

AA

DAT602

By Travis Byrman

Contents

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

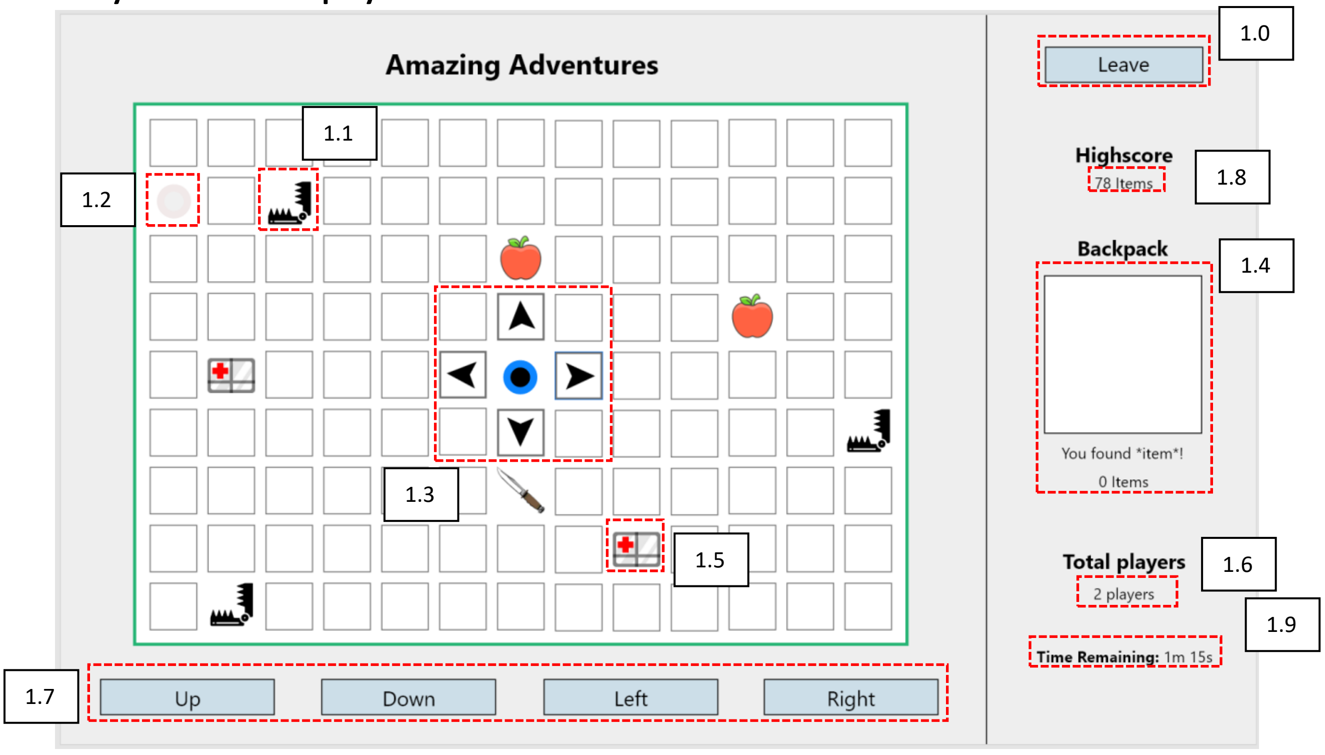
- After selecting the admin console, administrators will be shown a total list of players and active games.

- 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.

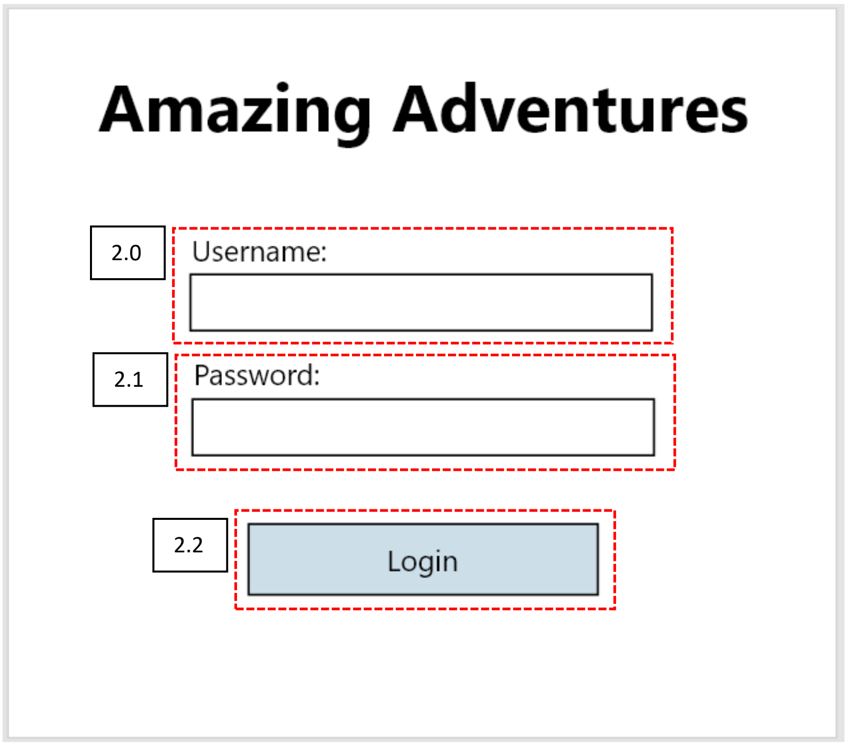
1.6 **Total player count** – This shows the total players in the game.

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

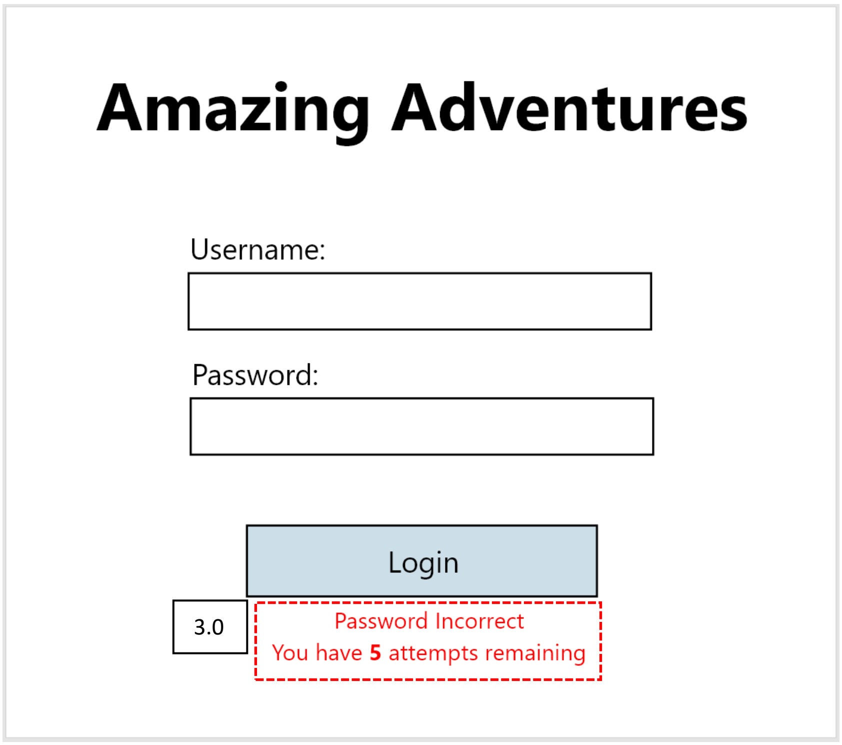


2.0 **Username field** – Input field where a user inputs their player account username.

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

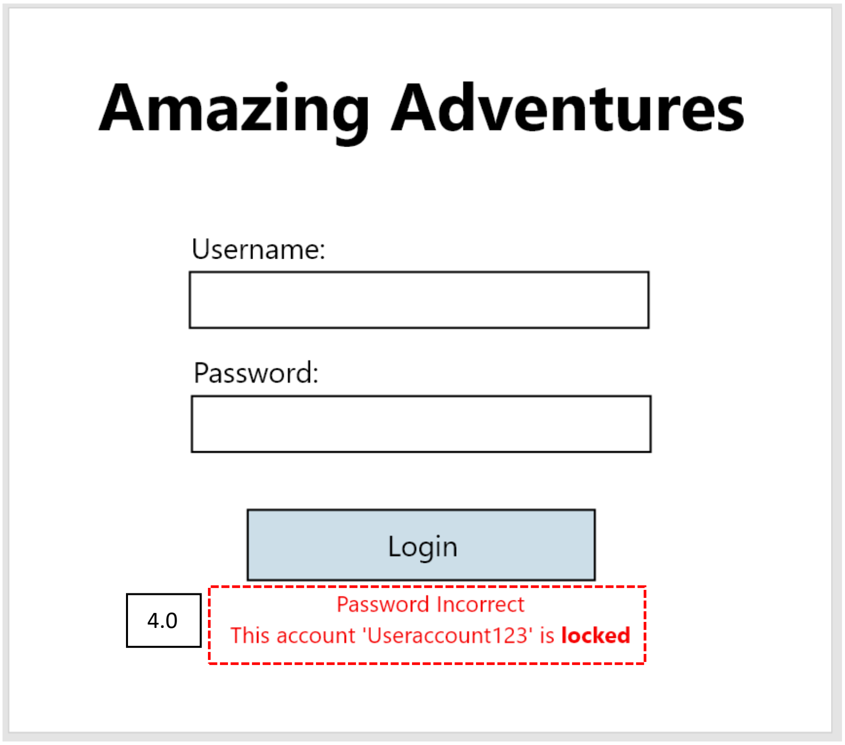
2.2 **Login button** – User clicks this button to login. If username or password is invalid login will be unsuccessful. If username is not a valid player account username, user is given the option to register. User has 5 login attempts before account is locked.

Storyboard 3: Login – Invalid Attempt



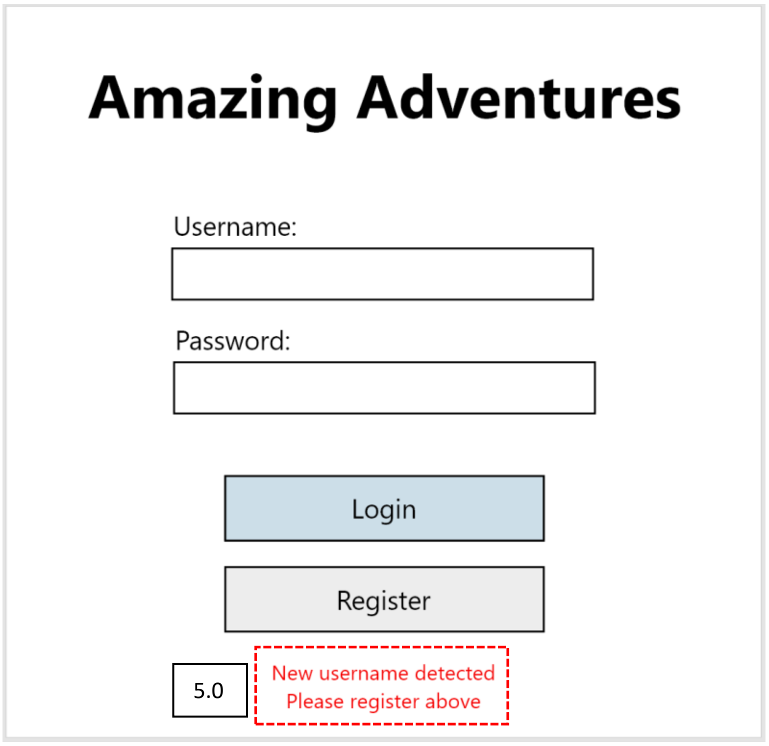
3.0 **Prompt** – Indicates user how many login attempts remaining.

Storyboard 4: Login – Account locked

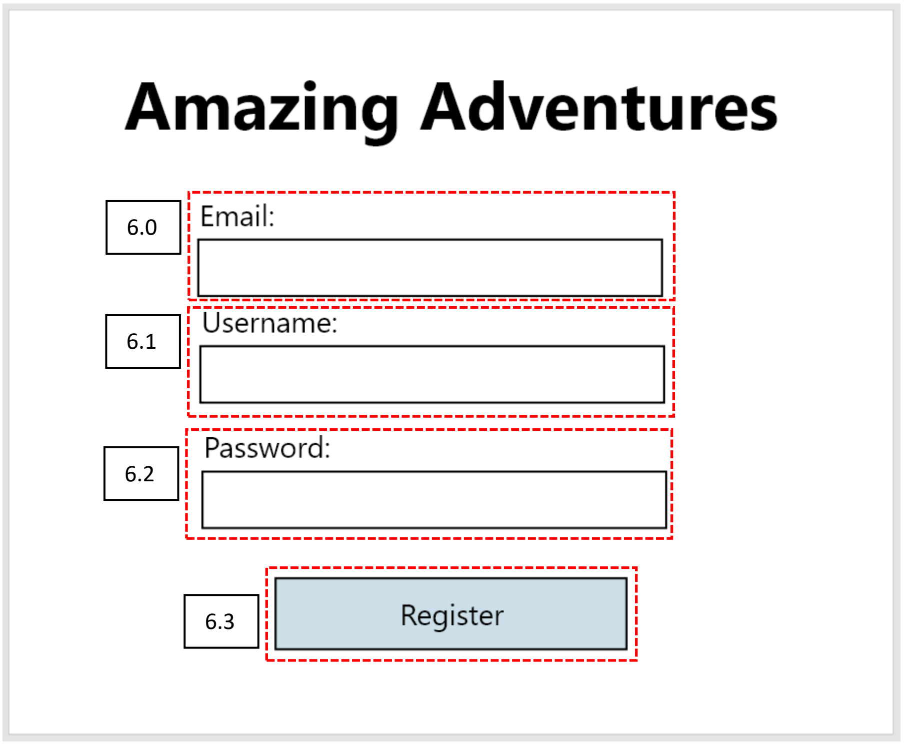


4.0 **Prompt** – Indicates to user that the account they tried to log into has now been locked.

Storyboard 5: Login – New Username Detected



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.



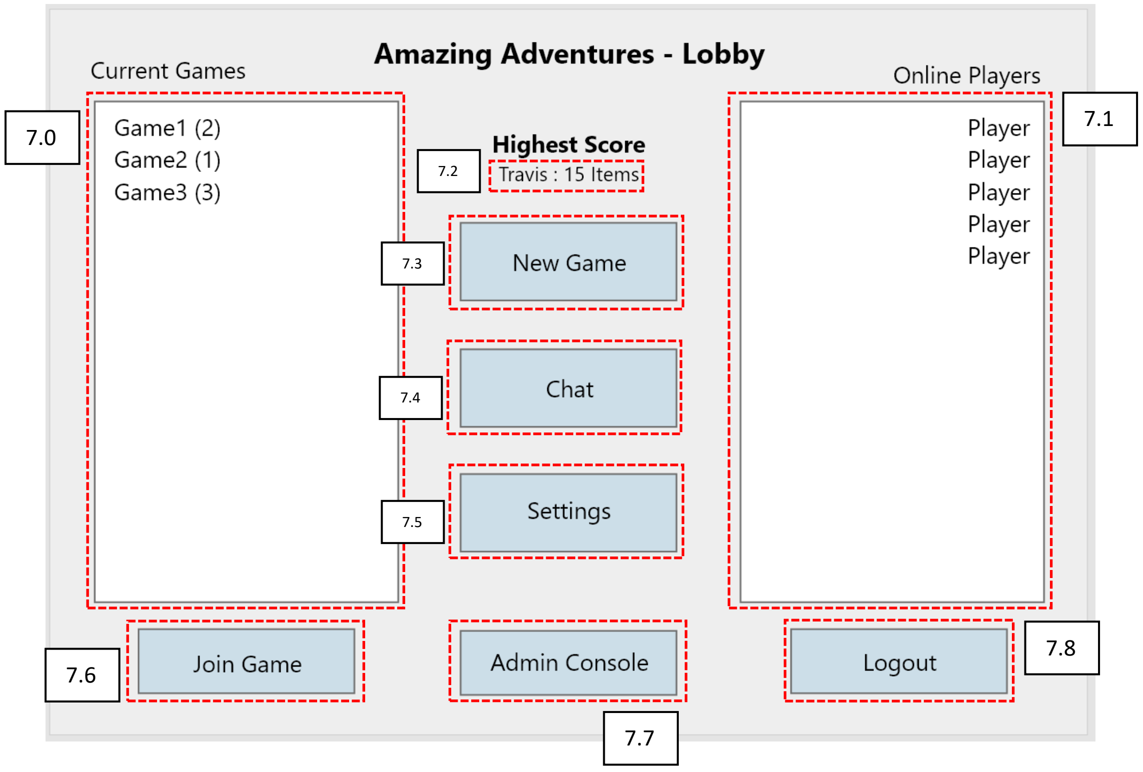
6.0 **Email field** – Input field where a user inputs their player account email address.

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.

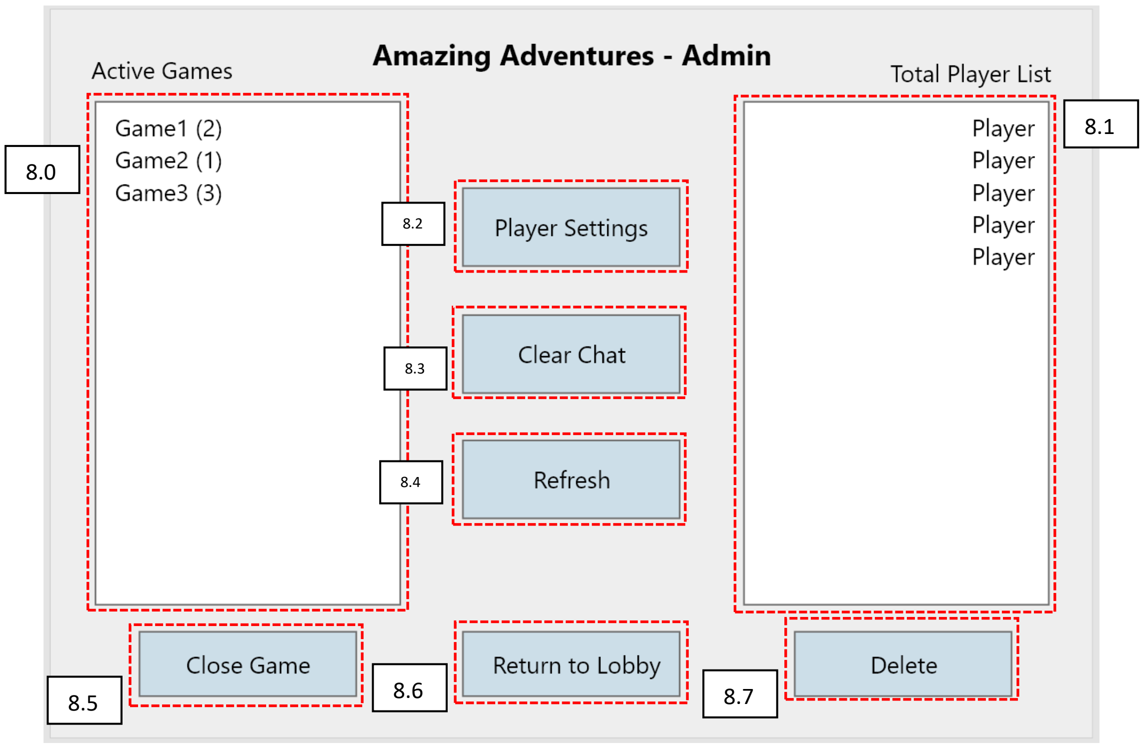
7.5 **Setting buttons** – Players can press this button to view their player settings.

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.

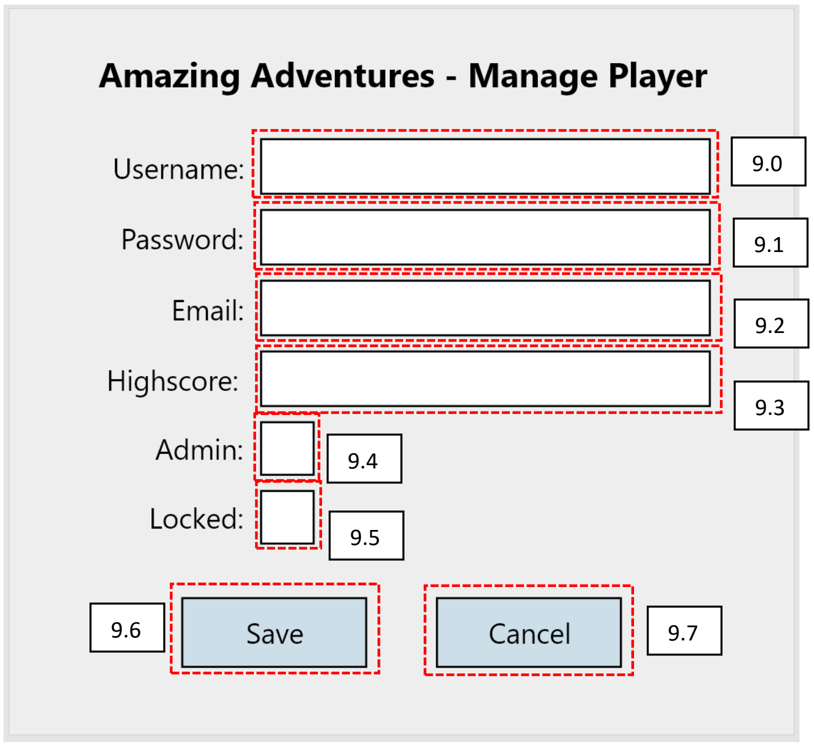
8.2 **Player settings button** – Administrators can press this button after selecting a player from the player list to view that players account settings.

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



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.

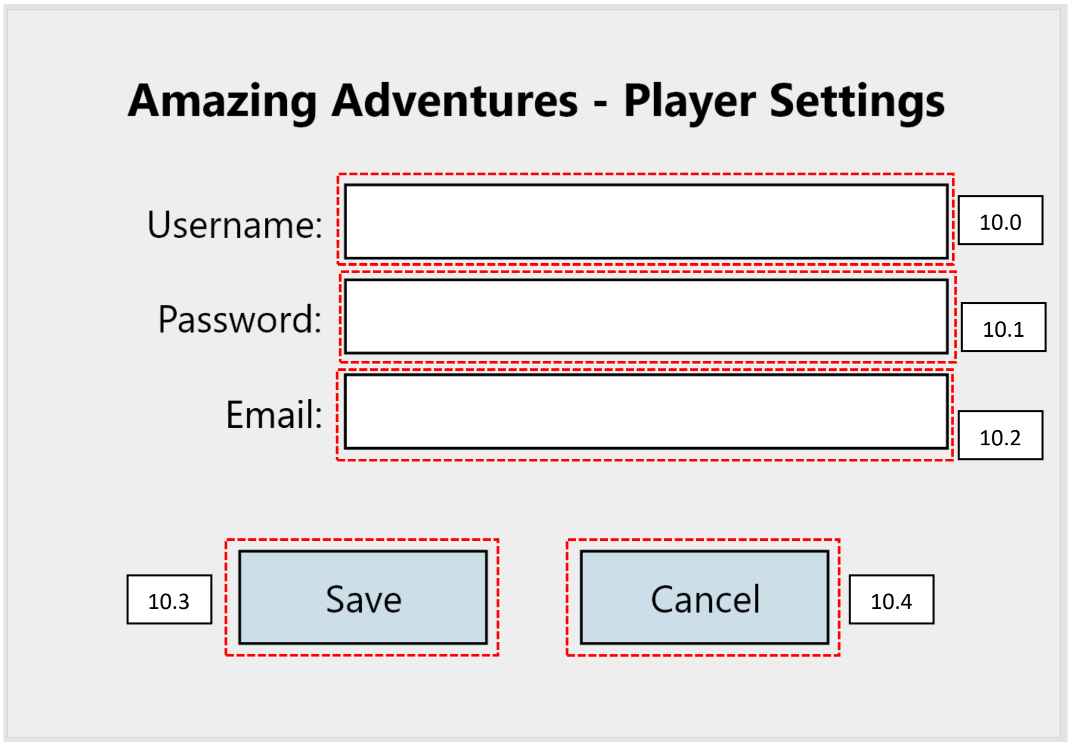
9.4 **Admin tick box** – Tick box field where a tick represents the user has administrative access.

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



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

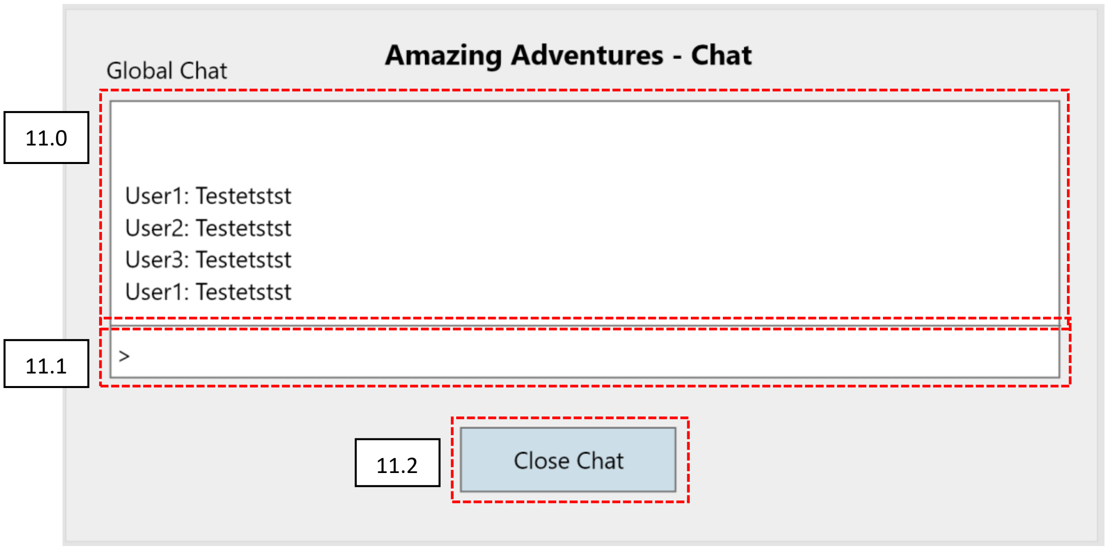
10.1 **Password field** – Input field where a user inputs their player account password.

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

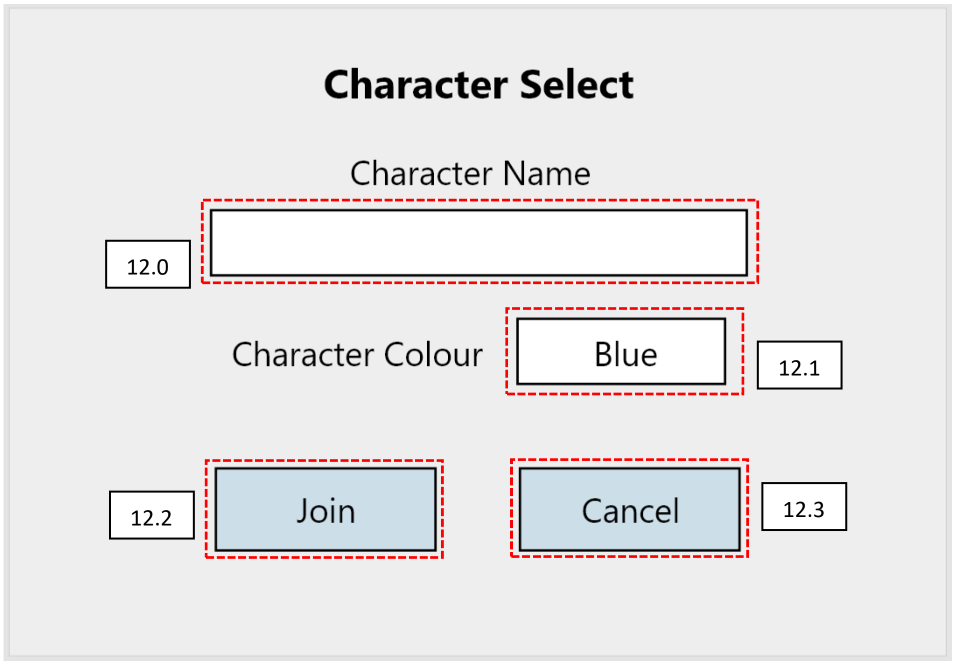


11.0 **Global chat text** – Players will see every player's text chat and date/time of publish.

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



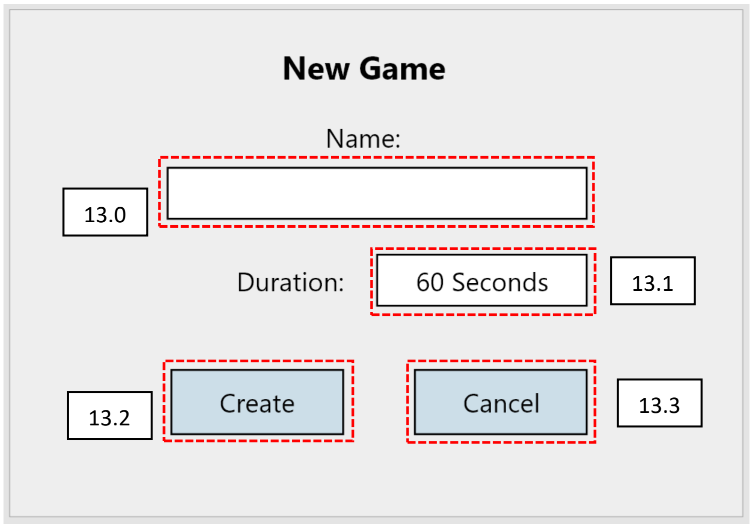
12.0 **Character name input box** – Input field where a player inputs their character name.

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



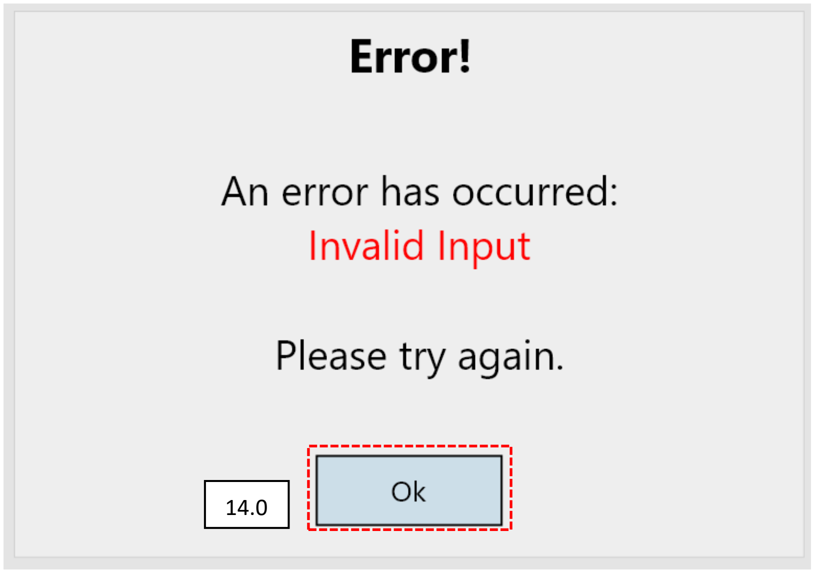
13.0 **Game name input box** – Input field where a player inputs their game name.

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



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.

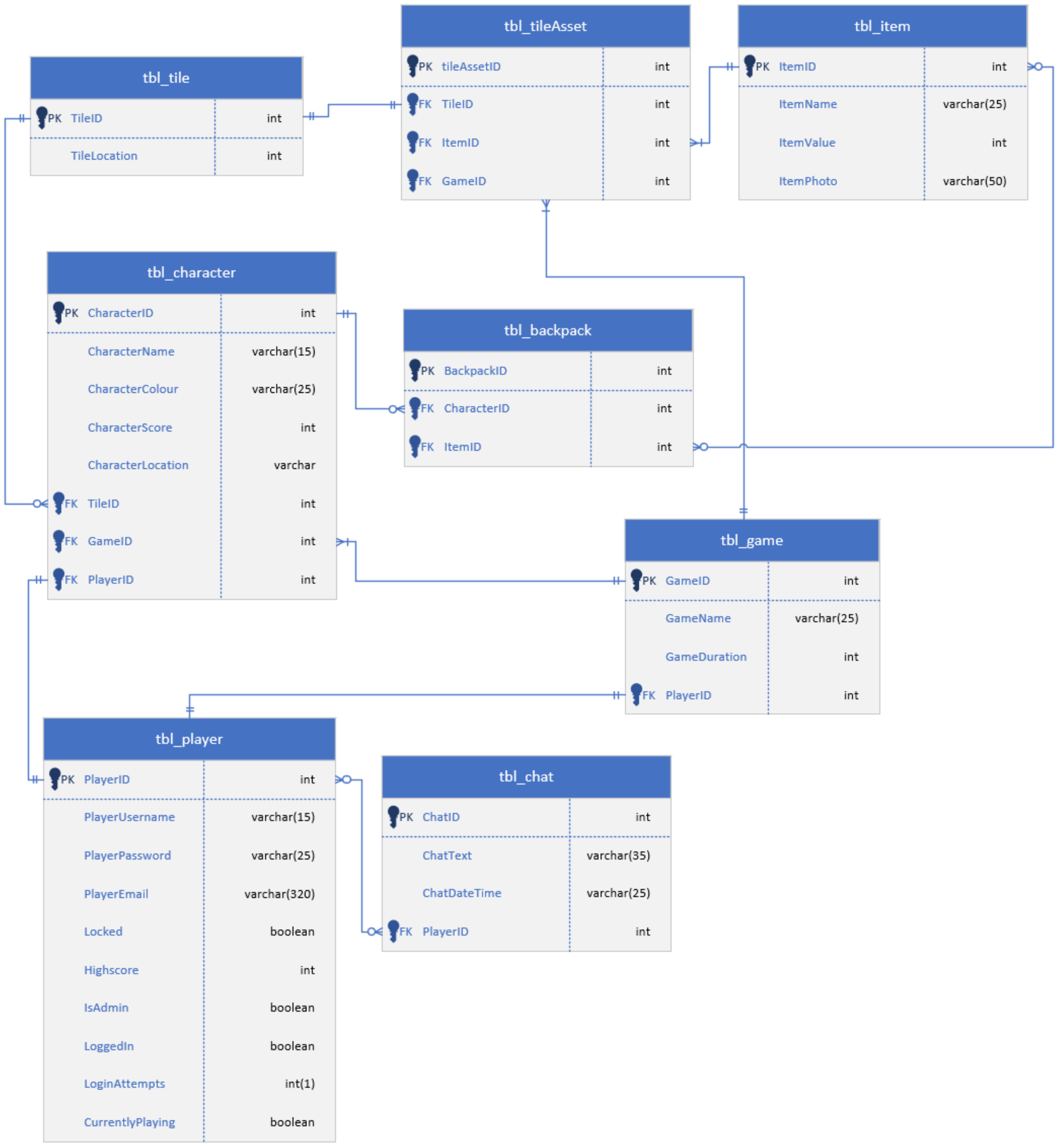
Design Choices

I choose 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 and 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: / 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 Collects item	Player Adds Chat Message	Player Changes Settings	Admin Creates Player	Admin Removes Player	Admin Locks Player	Admin Edits Player Info	Admin Ends Game
tbl_player																			
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 and account if they do not have an existing account.

After clicking the register button, a registration form will show, getting the users 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 unlock 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 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 players settings.

Create character

When a player wants to join a game, they will be prompted to create a character. Here they can choose their characters name and colour before entering. Once entered the game will be shown, and they 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 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 never 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.

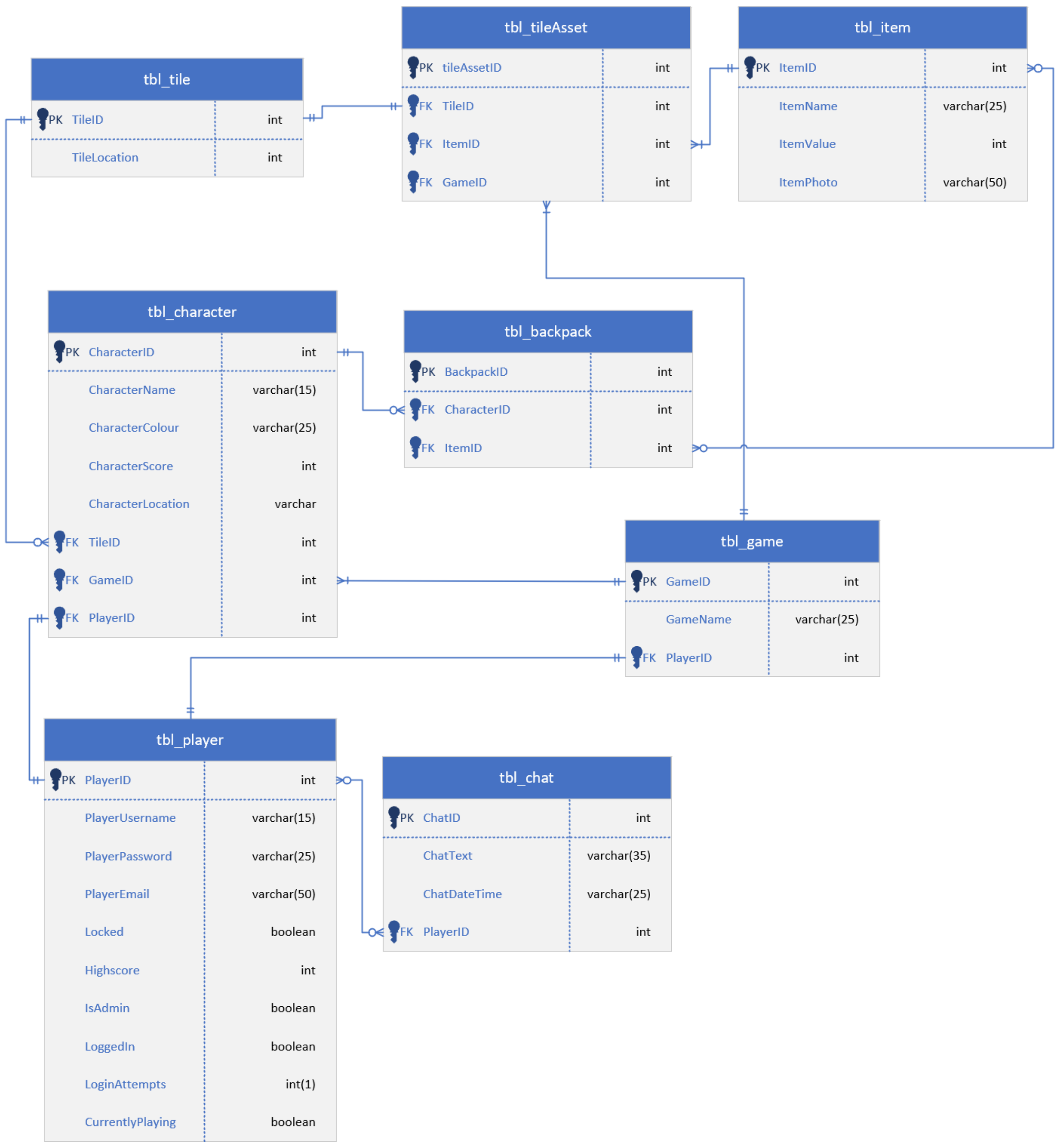
Multiplayer Support

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 dur 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 will 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 'addItems' 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.

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