Tile Game DAT602

Mark Christison

Nelson Marlborough Institute of Technology, New Zealand

Contents

[Contents 2](#_Toc44172934)

[Milestone 1 -Introduction 4](#_Toc44172935)

[Business Rules 4](#_Toc44172936)

[User 4](#_Toc44172937)

[Characters 4](#_Toc44172938)

[Game 5](#_Toc44172939)

[Map 5](#_Toc44172940)

[Administrator 5](#_Toc44172941)

[Item 5](#_Toc44172942)

[Mine 6](#_Toc44172943)

[Storyboards and Description 6](#_Toc44172944)

[Crows Foot Logical ERD 17](#_Toc44172945)

[Rational for logical ERD 18](#_Toc44172946)

[C.R.U.D Table 20](#_Toc44172947)

[SQL 21](#_Toc44172948)

[Usage Scenarios 21](#_Toc44172949)

[Milestone 1 - Conclusion 24](#_Toc44172950)

[Milestone 2 - Introduction 25](#_Toc44172951)

[SQL 26](#_Toc44172952)

[DDL 26](#_Toc44172953)

[Inserted Test data 26](#_Toc44172954)

[Update Statements 27](#_Toc44172955)

[Delete Statements 27](#_Toc44172956)

[Procedures/Transactions 27](#_Toc44172957)

[Register a User 27](#_Toc44172958)

[User Log in 27](#_Toc44172959)

[User Log off 28](#_Toc44172960)

[Edit User 28](#_Toc44172961)

[Delete User 28](#_Toc44172962)

[User Creates Character 28](#_Toc44172963)

[User Deletes Character 29](#_Toc44172964)

[Select Character to play game 29](#_Toc44172965)

[Change character to play game with 29](#_Toc44172966)

[Online Characters 29](#_Toc44172967)

[Create Game 30](#_Toc44172968)

[Leave Game 30](#_Toc44172969)

[C# Application 31](#_Toc44172970)

[Support for multi-player game play 32](#_Toc44172971)

[Atomicity 32](#_Toc44172972)

[Consistency 33](#_Toc44172973)

[Isolation 33](#_Toc44172974)

[Durability 33](#_Toc44172975)

[InnoDB 33](#_Toc44172976)

[Milestone 2 - Conclusion 35](#_Toc44172977)

[References 36](#_Toc44172978)

# Milestone 1 -Introduction

This assignment involves taking a game overview and producing a set of business rules, as well as storyboards with descriptions. From those developed rules and storyboards a logical entity relationship diagram(ERD) is produced. A set of stored procedures that are necessary for the operation of the game have then been listed, and a create read update delete (CRUD) table has be made to note where those changes will occur in the database. Finally a SQL file has been written to match the documented database. It includes Create, Update, Select and Delete statements which are there to test the functionality and ensure that the database has been built correctly.

The SQL file and CRUD table have been included in the submission document as additional files with this assignment submission.

# Business Rules

## User

* Enters username to log in
* If username is in system, finishes login by entering password
* Users are not required to register before playing, the system does that automatically. When a new username is entered, the system detects the new name and prompts the new user for a password and email. A warning is presented that they are about to start as a new user.
* When logged in, users can make one or more characters to play games with
* Has 5 attempts at logging in, otherwise is locked out and unable to log in until an administrator unlocks the account
* Has a total score made up of the sum of all character scores
* Can delete own user account

## Characters

* Move around 2D tile map by clicking on tiles/using key presses to move
* Move only to adjacent tiles
* Can only move onto a tile that is not occupied by another player unless it is the home tile on a map
* Collect assets from tiles
* Compete with other characters playing on the same tiled map
* The characters must have at least four and no more than six attributes that can be selected
* Can play against any/all other players
* When leaving a game, state of character is recorded
* When returning to a previously played game, starts on recorded tile unless occupied in which case the user selects an adjacent unoccupied tile to start on
* Can communicate with other characters through text chat. All chats are from one character to all other characters.
* Has a score for each game that is played.
* Selects a player to play a game with from a list of players who are not currently playing a game

## Game

* Can be played by 2 characters
* Can be played over a local network
* Displays other online characters and ranks them by their score, high to low

## Map

* Contains many tiles
* Has a home tile which can be occupied by many players
* There is only one map design that can be played on.

## Administrator

* Access an administrator interface
* Can kill running games
* Can add new users
* Can update data for current users
* Can delete current users
* Is sent an email if a user is locked out
* Can unlock locked user accounts

## Item

* Generated in a game
* Can be picked up by a character with the matching skill
* Used to gain points in gameplay on matching mine
* Has durability
* Durability decreases with each use

## Mine

* Generated in a game
* Used by a character with the matching skill and item to gain points

# Storyboards and Description

For the creation of the storyboards I have used Visual Studio. It has allowed me to create screen designs that while might not be the final screen design allow me to transmit the ideas that I have about the game. There are limitations to using this type of software to create these storyboards. The biggest being the time and effort associated with ensuring the quality of the designs. However, I have chosen this method as my art skills are rather poor and I feel that I am more easily able to transmit the ideas that I have about the game clearer using screen designs

A screenshot of a cell phone

Description automatically generated

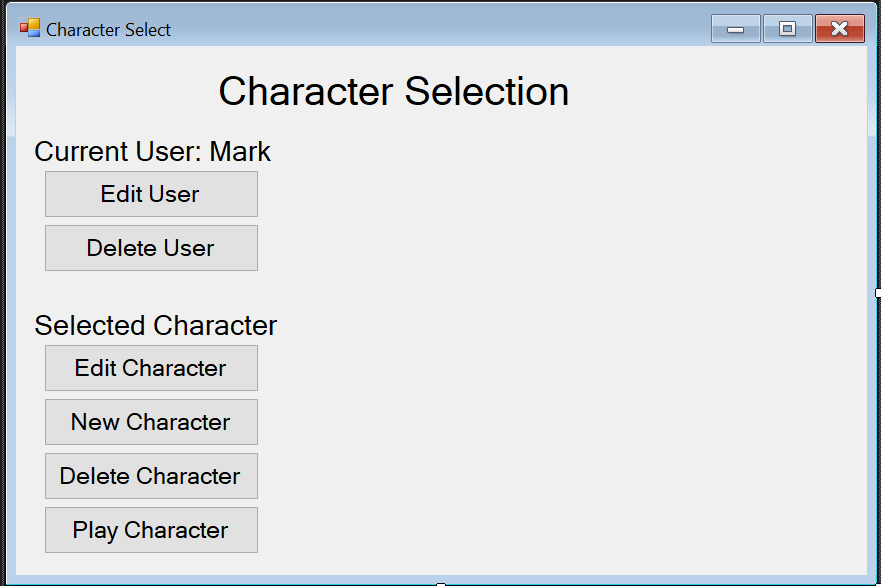
1. The User wants to play the game. They open the application and are presented with a window asking them for their Username.

The user enters a name to use in the game as they do not have an account already.

A screenshot of a cell phone

Description automatically generated

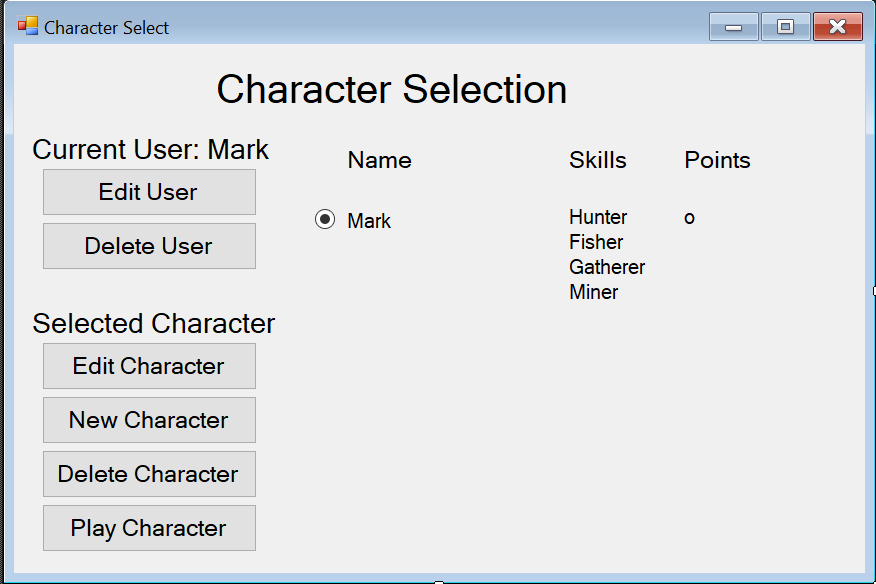
2. The user create an account with the entered username along with a password and email that are associated with the account. If at any time they do not wish to proceed the click cancel to be returned to the previous view else, clicks create account when the entered details are correct.



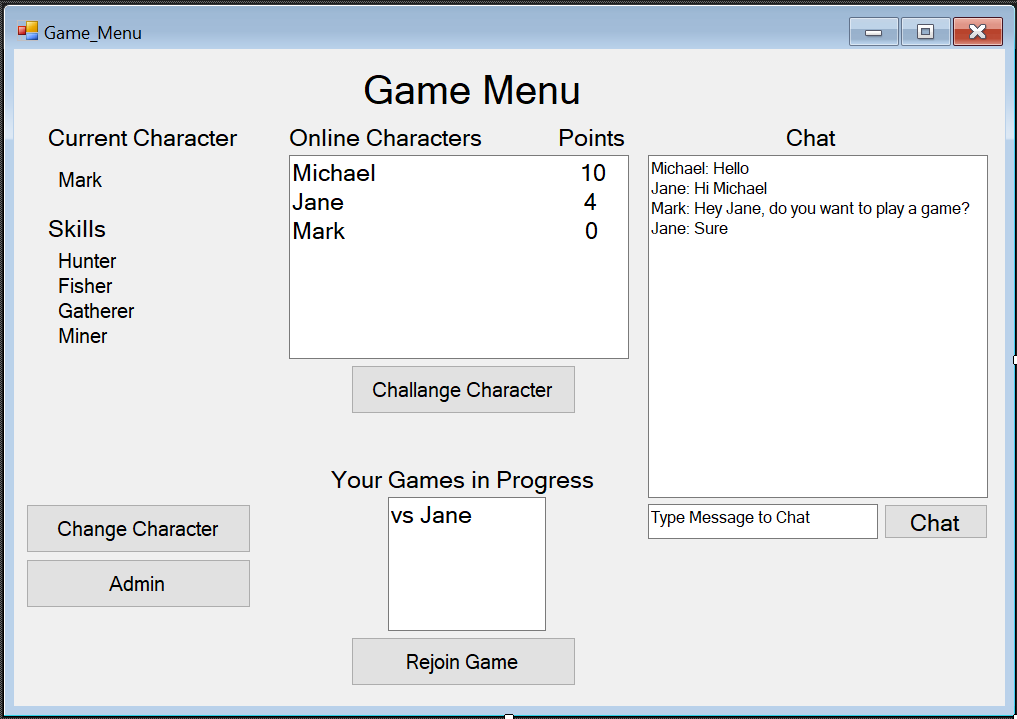
3. When the user has logged in, they are presented with a character selection screen. Since this user has no characters, they create a new character to play the game with.

A screenshot of a cell phone

Description automatically generated  
4. The user creates a character by giving it a name and 4 skills for it to use in the game as well as a colour to represent the player.

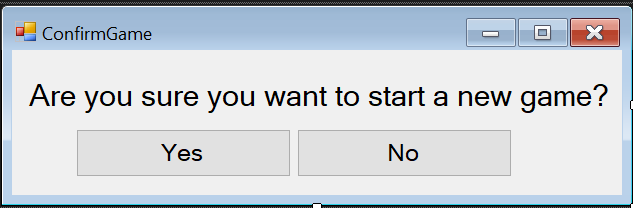


5. Once the character is made, it appears in a list of characters that the user has made. When the character is selected, a play button appears. When clicked the game menu is displayed. A character can be edited from this screen as well.



6. The game menu shows the Characters that are online and ready to be challenged in a game as well as any games the current character is already playing. If the user wants to change characters they can go back to the previous menu with “change character”. Otherwise they select a character from the “Online Characters” and challenges them to a game. If the player is an administrator, then they can access the admin menu from this screen too.

On the right side of the screen is the chat window. Characters can communicate to all other characters by typing in a message in the box and clicking chat.



7. When a character has been selected to play against, or when the current character has been challenged to play a game a confirmation box appears to check if the both characters want to play in a game.

A screenshot of a cell phone

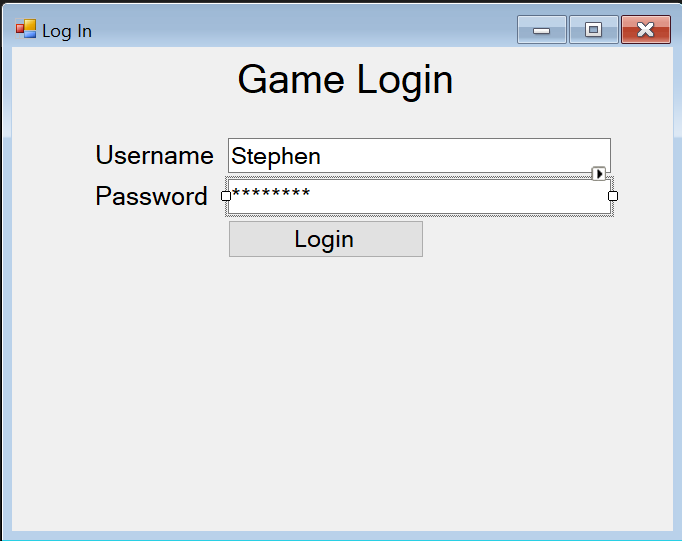
Description automatically generated

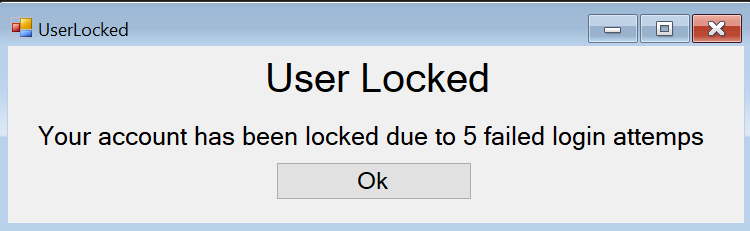
8. The Game state starts with both players starting on the ‘home tile’ of the map. Items that need to be collected and ‘mines’ (places to use the items on) are distributed on the map randomly. Characters must first pick up the items that match their skills and use them on the matching ‘mine’.

A screenshot of a cell phone

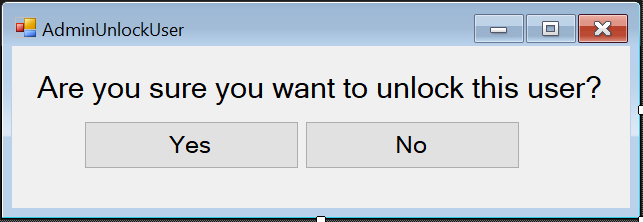
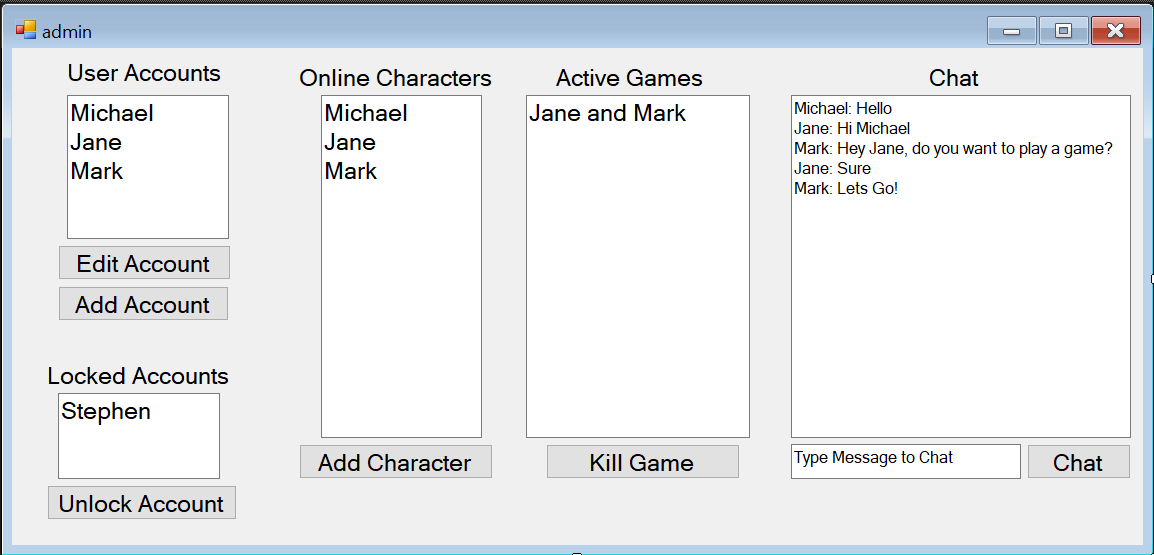
Description automatically generated

9. In the above game, Jane has picked up the bow and has used it on the target 2 times earning her 2 points. Mark has picked up the hammer and used it on the anvil 1 time. The score has been updated to show these changes. The items appear in the characters inventory and their quality degrades as they are used. Games procced until an administrator ends the game.





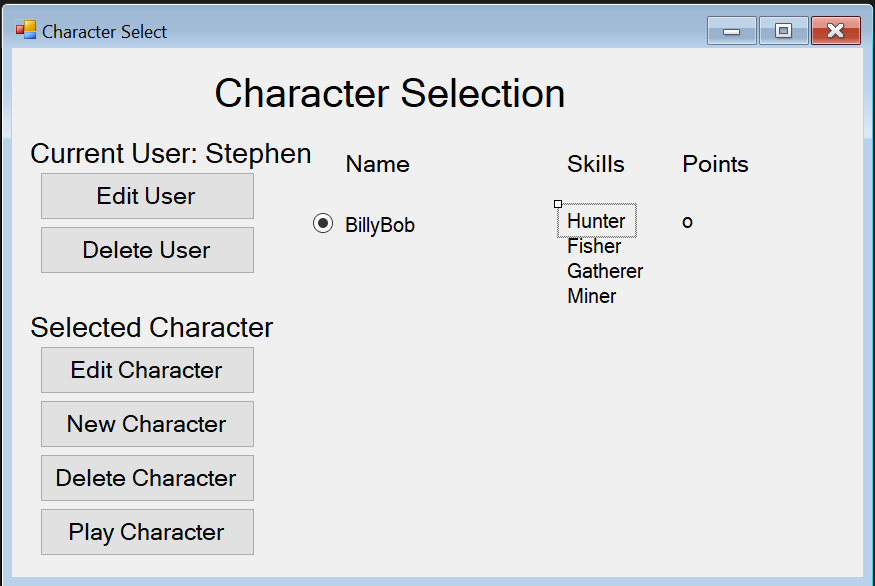
10. Stephen cannot remember his password to his account. He attempts everything he thinks it could be, but after 5 failed attempts he is locked out of his account. Each time he tries to log in after this attempt he is presented with the same “User Locked” window

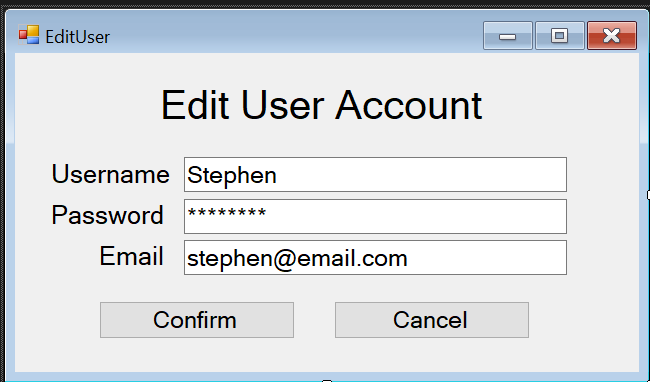


11. Michael is an administrator. From the game menu he clicks the ‘admin’ button which takes him to the admin menu. From here he can kill running games and see a list of the online players, edit characters, edit users and add users. He wants to unlock Stephens user account. He unlocks Stephens account by selecting his name from the list of locked accounts and clicks unlock account. A confirmation window appears, and he selects yes to confirm his choice. A similar confirmation occurs for all administrator actions.

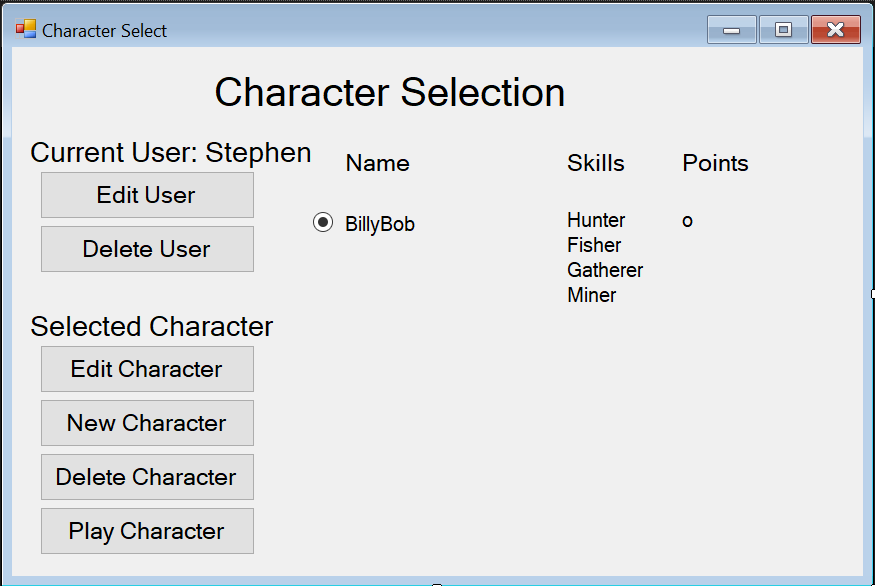
D

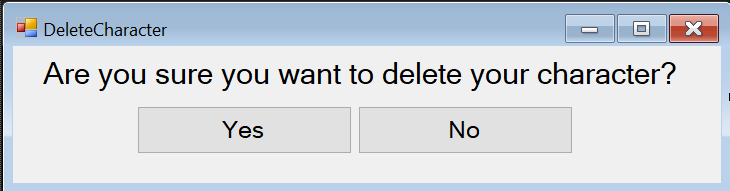
12. Stephen wants to log into his newly unlocked account. He enters his username and the system detects that he is a current user. The form displays the password field and login button. He enters his password and clicks login.

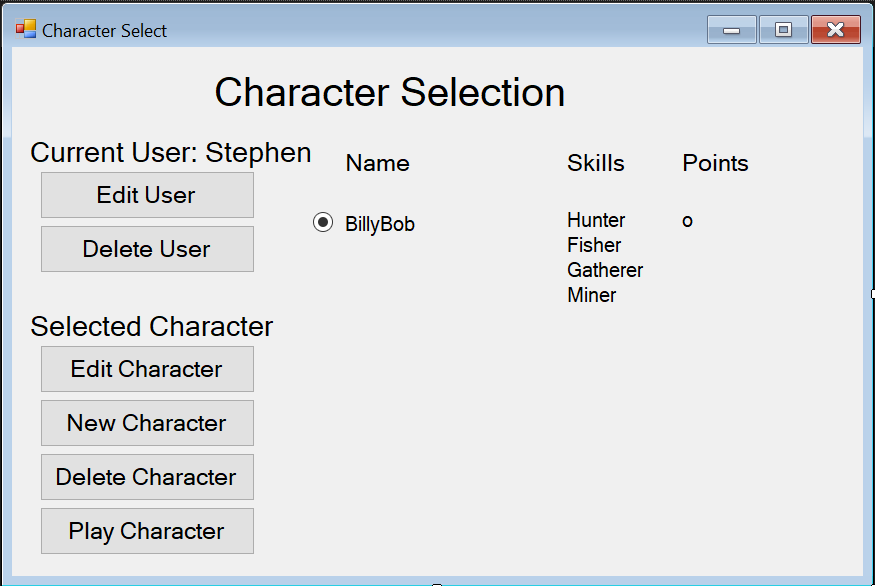


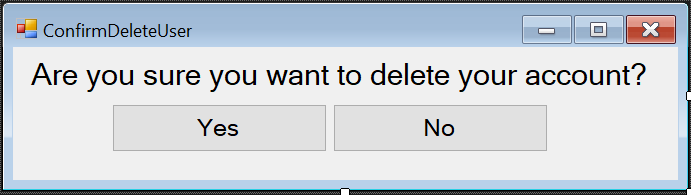


13. Stephen wants to change his password to something more memorable so that he does not get locked out of his user account again. He clicks edit user and is presented with the Edit user screen. He changes his password and clicks confirm to record the changes to his account details.

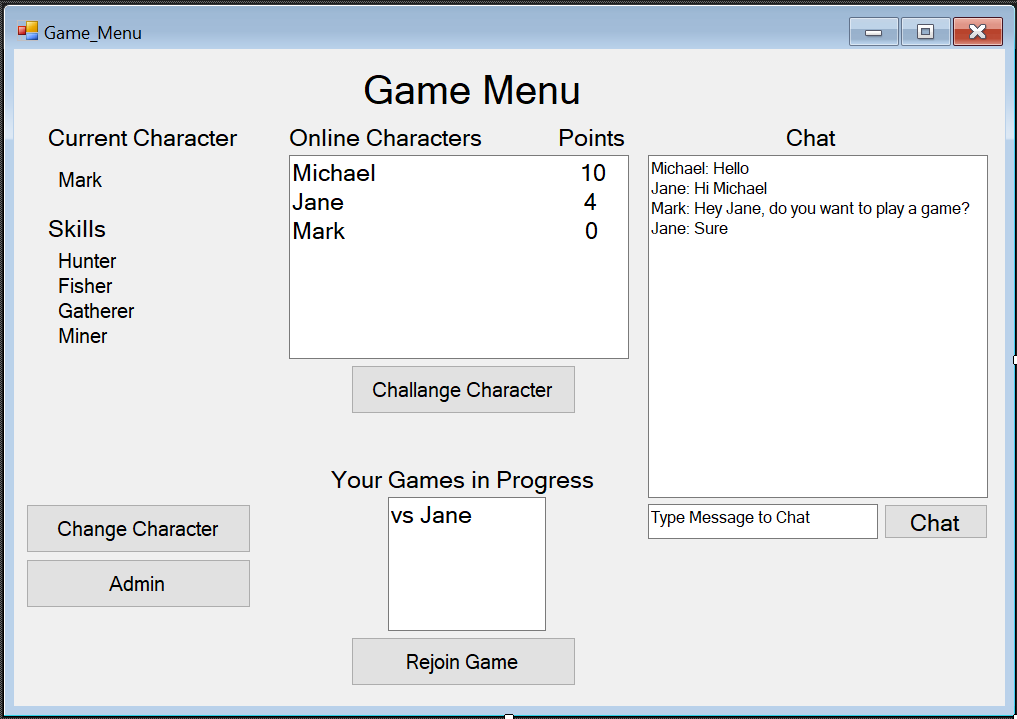


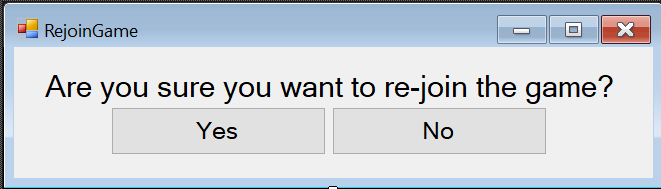
  
14. Stephen no longer wants to keep the character ‘BillyBob’ so from the character selection screen he clicks on the character and selects ‘Delete Character’. A window asks for confirmation of deleting the character. He selects yes and there character is deleted.





15. Stephen decides that he no longer wants to play the game at all. He logs into his account and clicks ‘Delete User’. A confirmation pop up appears prompting “Are you sure you want to delete your account?” and he selects yes, and his account is deleted.





16. To re-join a game that a character leaves, from the game menu they select the game that they want to re-join and click ‘Rejoin Game’. A window appears to confirm the choice of the user.

# Crows Foot Logical ERD



# Rational for logical ERD

I started making this model from tables with only primary keys. Those tables are User, Skill, and Map. Then I made the tables that had foreign keys to the tables already created. Finally I made all the join tables that solve the many to many relationships that cannot exist in the logical entity relationship diagram.

The user table first was made first. Each user can have zero or more characters. Each user will be identified by a unique number as a userID. Each user has one username, email and password. The email and password are used to log into their account. Their account records the times that they have attempted to log in to check if the user needs to be locked out. The userScore is cumulative for all the characters that user owns. The isLocked attribute is a binary yes/no and is set accordingly, with the default being not locked. The isAdmin attribute is also a binary yes/no for if the user is an administrator. The isOnline attribute is also a binary yes/no to indicate whether the user is online or offline.

I next made tblSkill. Each skill relates to only one item that it matches and it relates to zero or more characterSkills, the join table between character and skill. tblSkill has the attributes skillName, as a identifier for each name the skill has and a skillDescription which is a description of that skill.

I then made tblMap. It has a unique identifier of mapID, a home tile location which is recorded with x and y axis attributes, as well as x and y size integers. It relates to zero or many rows of tblcharacterMap and 1 or many Tiles.

TblItem has the attributes of itemID which uniquely identifies each row, a mactchingSkill foreign key which pair the skill to the item that the character to use, an itemName for the name of the item, an itemDurability which is that row’s item current durability and a maxDurability which is the starting or original value that the durability can be. Each row in the tblTileItem relates to one and only one row from tblSkill, one and only one row from tblMine, zero or many rows in tblTileItem, and zero or many tblCharacterItem rows.

Tblmine has the attributes mineID to uniquely identify each row, a foreign key to tblItem called minedBy and a mineName which is the name of the mine. Each row on table mine can relate to zero or many rows in tblMineTile and one and only one row in tblItem.

Tbltile has the attribute of itleID to uniquely identify each row, a foreign key to tblMap called mapID, and an x and y location integer to identify where it is located on the map it is part of. Each row in tblTile relates to one and only one row in tblCharacterTile row and tblMap. As well each row in tblTile relates to zero or more rows in tblMineTile and tblTileItem.

TblCharacter has the attributes characterId to uniquely identify each row in the table, character name which is the name of the character, userId which is a foreign key relating to tblUser, an x and y position integer value to record where the character was last on a map, characterScoreTotal which is the acculimated score for that character across all games, and isActive which indicates if the character is active or not. Each row from tblCharacter relates to exactly 1 user who owns the character, zero or many character items which are the character inventory, 4 character skills (denoted 1 or many) which are the skills the character has, zero or many chats, zero or many characterTiles and zero or many characterMaps.

TblChat has the attributes chatID which is a unique identifier for each row, sender which is a foreign key relating to the character which sent the message, and a message which is the string of characters sent to the database by the user to be displayed to all characters. Each chat relates to exactly one character who sends the chat.

TblCharacterSkill has the attributes characterID which is a foreign key to the character and skillID which is a forgein key to a skill. Each row in this table relates to only one character and only one skill.

TblTileItem has the attributes tileID which is a foreign key to a tile and itemID which has a foreign key to an Item. Each row in this table relates to only one tile and only one item.

TblCharacterItem has the attributes characterID which is a foreign key to a character and itemID which is a foreign key to an Item. Each row in this table realates to only one character and only one item.

TblCharacterMap has the attributes characterID which is a foreign key to a character and mapId which is a foreign key to a map. Each row in this table relates to exactly one character and one map.

TblMineTile has the attributes mineID which is a foreign key to a mine and tileID which is a foreign key to a tile. Each row in this table relates to exactly one mine and one tile.

TblCharacterTile has the attributes characterID which is a foreign key to a character and tileID which is a foreign key to a tile. Each row in this table relates to exactly one character and exactly one tile.

# Usage Scenarios

Register a User

Refers to storyboards 1 and 2. New user to the system creates an account.

User Log in

Refers to storyboards 12. A user with an already registered account logs into the system.

Edit User

Refers to storyboard 13 From the character selection screen, a user can choose to modify their account.

Delete User

Refers to storyboard 15. From the character selection screen, a user can select one of of their characters and choose to delete it.

User Locked out

Refers to storyboard 10. If a user has 5 or more failed log in attempts, they are ‘locked out’ of their account and require an administrator to unlock their account.

User Creates Character

Refers to storyboard 4. A user creates a new character that is associated with their user account.

User Deletes Character

Refers to storyboard 14. A user chooses to delete their character associated with their user account.

Select Character to play game

Refers to storyboard 5. The logged in user selects from a list of characters to play the game with.

Create Game

Refers to storyboard 6. Once the user is logged in and has selected character they can choose or be chosen to play a game.

Leave Game

Refers to storyboard 9. During game play a character can leave the game, saving the state of the character in the game and returning to the game menu.

Game Generates Items

Refers to storyboard 8. When the game is generated, the items inside the game are also generated.

Game Generates Mines

Refers to storyboard 8. When the game is generated, the mines inside the game are also generated.

Return to Game in Progress

Refers to storyboard 16. From the game menu when a player has left a game in play they can return to that game

Character makes Move

Refers to storyboard 9. During game play, a character clicks or presses a key associated with a movement to another tile.

Character picks up item

Refers to storyboard 9. During gameplay, if a character is standing on a tile with an item and they take the appropriate action to pick up the item (possibly a key press) it is added to their inventory.

Player Uses Item on Mine

Refers to storyboard 9. During gameplay, if a character has an item that matches a ‘mine’(place to use items) and the matching skill, they can ‘use’ the item (with a keypress). Using the item results in an increase in points and a decrease in the item’s durability.

Character makes a chat

Refers to storyboards 6, 8, 9, and 11. Once a character has been chosen for game play, in gameplay or in the admin panel, a character can chat with all characters.

Access Administrator Interface

Refers to storyboard 11. A character who is an admin can from the game menu access an administration window where they have administration rights over the games, characters and users.

Admin Kills Running Game

Refers to storyboard 11. An administrator can kill any/all running games.

Admin adds a user

Refers to storyboard 11. An administration can add users to the database.

Admin edits user

Refers to storyboard 11. An administrator can modify any user account in the database.

Admin deletes user

Refers to storyboard 11. An administrator can delete any user account in the database

Admin unlocks locked user

Refers to storyboard 11. An administrator can unlock any currently locked user accounts.

# Milestone 1 - Conclusion

Through the process of this assignment I have developed a set of business rules, created a set of storyboards about how users will interact with the game, developed a logical ERD CRUD table showing all the stored procedures required for the game, a SQL script used to create the game, and an explanation about the usage scenarios.

This project has given me clearer understanding of the relation of a logical ERD to SQL. I can see how the next steps of database creation will happen in the implementation of the procedures that have been identified in the usage scenarios and further documented in the CRUD table.

# Milestone 2 - Introduction

This document carries on the work done previously in Milestone 1 where the business rules, storyboards and an outline for the procedures need to be created where laid out. This milestone revolved around completing the SQL required for making the database schema and the procedures needed for the game play. To ensure the correctness of the database and functionality of the build, a set of test data has been included in the SQL. This serves functionally as it provides the initial game dataset.

After the creation of the database was completed, a C# console application was created to interface with the database and test its functionality. The app has been written to accept dynamic input as to allow for complete testing of the database and to catch any and all possible errors in the code.

Additionally, as transactions are being used to interact with the database, note has been taken to ensure that those transactions have atomicity, consistency, isolation, durability (ACID) in the case of any errors or disasters occurring to the database.

# SQL

## DDL

In creating the procedures needed for the usages scenarios I have realised that the database requires a few changes. Most of the changes have happened to the primary keys on several tables DDL. While initially manually incrementing each data row that was entered to the table was fine, I knew that this needed to be smarter for adding extra data especially in game play. While it was an initial consideration to give these tables an ID column that was an integer, I have instead opted of more natural keys in the data.

The user table has a natural key in the username. Understandably, this might slow down the database if there were to be a large number of rows in the table, however, the size of the database for this problem, it should not grow large enough for that to become an issue. However, I do note it as a potential issue in the future that may require changes.

As with the user table, the skills table, item table, mine table, map table, and character table all have used the attribute that “names” them as a primary key.

An auto incremented integer primary key has been added to the chat table as there was no other logical way to reference every row from the data in the table already.

The other changes to the DDL include the addition of “On update cascade” and “on delete cascade” to the foreign key columns in several tables. This is to support the changes that could occur to the data across the database and to not orphan any data. An example of this is in table character on the column username. As one of the usage scenario’s is for a user changing their name, if the username column was not updated when the primary key in the user table was, then the data rows with the old user name would no longer reference the new user and all old data would be orphaned. Likewise, if a user deletes their account, all the characters that user had will be deleted as they are no longer needed by the database.

## Inserted Test data

To meet the new DDL, the test data has been updated. New test data has been included as well to more accurately reflect the game that is being created. All the concepts from the draft in milestone 1 have been fleshed out in the database. All game assets have been created and a set of different maps has been created.

The inserted data ensures that the tables still have the initial integrity that they had during the tests carried out for milestone 1.

## Update Statements

The initial update statements where intended as tests only and have therefore been commented out. The newly created procedures/transactions now encompass all the required updates for the game.

## Delete Statements

The delete statements have been removed as the created database now has a functional set of data and the deletion of the data would disable some of the functionality, such as the administrator procedures. As with the update statements, these have been commented out.

## Procedures/Transactions

The original usage scenarios as defined in milestone 1 have provided a template for constructing the procedures. Notable, several usage scenarios have been reduced from 2 or 3 scenarios to 1 procedure. As well, other newly discovered scenarios have been created into procedures that the game will require. Some of the scenarios have been split into 2 procedures to allow for user input as needed.

These transactions are similar in a sense to use cases. As such, while the do not follow the normal structure of a use case description, I will only talk about the success case for each scenario.

## Register a User

SQL Lines 400-424.

This procedure checks to see if the username or email passed as parameters already exist in the database. If not then this procedure inserts the parameters into the user table. It also catches exception cases where a username or email are already in the database.

## User Log in

SQL Lines 428-480.

This procedure initially sets variables locally from the database which match the same data row from the user table as the username parameter.

If there is a matching user in the database as the parameter, the password passed as a parameter is checked against the username data row. If that is the same, then a check is made on the Login in attempts count and the locked status of the user. If the user has less that 5 login attempts and the user is unlocked and their password matches the stored password, then they are logged in. Their online status is set to true.

The user locked out scenario from milestone 1 has been amalgamated into this procedure. Each failed log in attempt with a username that exists in the data base increments a counter. Once that counter hits 5 or more then the user row updates on the column is locked and the user is unable to log in, even if on future attempts their password is correct. An administrator must unlock their account.

## User Log off

SQL lines 484-503

This procedure can only be access if a user is already logged in from the app. It updates the user table and sets the online status to false.

## Edit User

SQL lines 507 – 542

This procedure checks if the current username parameter exists in the database, if it does it then checks if the new username parameter does not exist in the database. If it does not exist it then checks that the email parameter does not exists in the database. If all the checks return true, then the user table is updated on the row matching the username with all the input parameters.

## Delete User

SQL lines 546 - 567

If the username passed as a parameter exists in the user table then that row is deleted. Notably, this procedure can only be accessed once a user has logged into their account, thus not requiring other parameters as checks.

## User Creates Character

SQL lines 571 – 600

If the character name passed as a parameter does not exists in the character table, then an insert is done on the character table. Also, all the skills passed as parameters are inserted into the character skills table.

## User Deletes Character

SQL lines 605 - 649

This scenario is broken into 2 different procedures. This is because it requires input from the user based on data from the data base. The first procedure that is run returns all character names from the character table which match the username of the user.

The user then selects the character name that they want to delete. A check is done to make sure that this character exists and belongs to the player then it is deleted.

## Select Character to play game

SQL lines 653 - 679

This scenario uses the first procedure from the previous scenario. First it gets all the characters that belong to a user from the user table. Then based on the character name that the user selects, updates the character table where the character name matches the parameter character name (the one the user selected).

## Change character to play game with

SQL lines 683 - 709

This procedure is a new procedure that I have identified that adds extra functionality. A user can change their character instead of being forced to play only with the same character once logged in.

When this procedure is run, the database updates the character table and sets the active status of the character name matching the parameter character to false.

## Online Characters

SQL lines 713 - 730

This is another new procedure. This will show all the online characters that can be challenged to a game. This procedure will be used solely for the purposes of the GUI version of the game.

This procedure returns a set of character names who have the is active column set to true.

## Create Game

SQL lines 734 – 874

This scenario encompasses several of the originally identified scenario’s, game generates items, game generates mines, and create game.

This scenario requires 3 procedures. The first gets the online characters whose character name does not match the character name of the current character.

The second procedure gets all the maps that are available for play.

The third procedure creates the game. Based on the choice of map from the user, local variables are set to the user’s choice of map. Then the database checks to see if either the current character or the character chosen to play against are already playing a game. If they are not, then the game can be made. Both characters get inserts that create rows on the character map table where the character name and map name are the same as the parameter. The database then “spawns” the items and mines inside the game, on tiles that that map has. Check are made to ensure that mines do not spawn on the same tiles as items. Characters are placed on the home tile of the map. Inserts are made to the character tile table, putting both characters on the home tile of the map.

## Leave Game

SQL lines 789 – 926

Refers to storyboard 9. During game play a character can leave the game, saving the state of the character in the game and returning to the game menu.

After attending class on 8 June 2020 I realise that this might be too much for the requirements for this assignment. As such, I have concluded to not complete the logic for the rest of the procedures. However, they do follow similar lines of thinking.

# C# Application

The application to interface with the data base has been written in C#. The choice for this is because C# is an incredibly slick and modern language. The application has been made to accept dynamic input from the user through a range of menus. These menus can be navigated through using the command line(CLI).

By using an application that is dynamic, the CLI application more accurately represents what will be needed from the application from the 3rd milestone. As nothing is hard coded into the main class, testing can be more in-depth and realistic data can be added to the database through the app. As well, it more accurately represents how the GUI version of the game will be interacting with the game. Also, just more generally, it feels simply better not hard code in tests, as that what you are only testing what is hard coded in. Its also harder then to catch errors or exception cases.

Along with the standard packages that come with a C# console application, I am using the MySQL Data package that has been included from the NuGet package manager. This package provides methods for creating connections to a MySQL database and other useful methods for interacting with a MySQL.

The application has 2 classes. The main class contains all the input menus which the user interacts with. I would have liked to split this off into different classes if I had the time, however this has not happened. The second class is the test class which contains all the methods for making calls to the connected MySQL database.

The biggest benefit of breaking up the program into several different classes is that it allows code reuse. The test class will be able to be reused in milestone 3 for the GUI version of the game.

# Support for multi-player game play

To ensure that the database can support handling many requests from different users at or near the same time, all of the procedures have been structured as transactions. Transactions in SQL have 4 main properties that ensure that the database will work correctly even in the event of a power failure or other fault event or failure.

There are 3 main types of failure that can occur. A transaction failure is caused when there is a improper input to the database or a violation of consistency occurs. A system failure is when either the host or server connected to the database suffer a critical error and the operating system crashes. A media failure is when either the primary or secondary storage of

These 4 properties are knowing by the acronym ACID and are:

* Atomicity
* Consistency
* Isolation
* Durability

## Atomicity

Inside each transaction there can be many statements. If each of these statements where to run normally, they would execute sequentially for example selecting data, inserting new data, updating a table. If on the third query there was an error in the database, then only that query would not execute correctly.

However, Atomicity ensure that in all transactions it is either all or nothing. All transactions inside the query must execute without error or warning (or else exit with a handler for this conditions) or else the transaction will not be parsed.

In MySQL, this is handled by the adjusting the auto commit setting. Normally, this setting is true, meaning that every query is executed straight away. However, when a transaction is started, the auto commit setting is set to false. Only the Commit statement makes all the changes in the transaction permanent.

If the transaction encounters an error, the transaction can rollback to the previous state of the database before the transaction started. This cancels the changes that are made in the transaction.

## Consistency

Consistency ensures that the database is taken from a known and valid state, to another valid state. Transactions must only effect the data stored in allowable ways. If invalid data was to be entered in the database ie not following a constraint, then there would be issues in the child or parent tables of the rows entered in. Ensuring that data is input correctly into the database allows the same or other transactions or queries to be executed by the database engine without incurring fault.

If for whatever reason, there was data the was entered that was invalid then the database again would rollback to a consistent state.

## Isolation

Isolation ensures that only one transaction can occur at once. If 2 or more transaction where to try and interact with the data in a database, one transaction might delete or updated the data the another had already changed or deleted. To resolve this, the database management system will queue the transactions so that are preformed one after another. That way, only one transaction can interact with the database at once. However, to the transaction, it seems as though it is interacting with the database as it normally would as though no other transactions or users where also interacting with the database.

## Durability

Durability ensures that after the transaction has been committed, the changes made by the queries remain. While the database will use volatile memory in ram for temporary storage, durability enforces that once transactions are completed the changes are stored to secondary storage.

### InnoDB

The database engine that MySQL uses called InnoDB ensures that all transactions follow the ACID model. It includes features like Commit which ends a transaction, and rollback which undoes all the changes made by the transaction. It also includes a crash recovery feature. When a crash occurs, any queries executed outside of a made outside of a transaction are completed and saved. Any query executed inside of a transaction that does not commit is rolled back.

For this milestone, the database uses transactions that are managed by InnoDB for all user input queries that are executed. As such, the database engine enforces ACID on all the transactions that take place.

# Milestone 2 - Conclusion

This assignment was possibly one of the hardest yet most rewarding pieces of work that I have ever done. The structuring of first the SQL, then test the SQL, to writing the application that interfaces with the built database, to testing of the database was an elegant process to perform.

All the procedures required for the game to run have been built. I suspect that I may identify more procedures the need to be built for the creation of the GUI version of the game as required in milestone 3, however I am quite confident that I have captured most of the requirements for the full game.

The application in C# works effectively accepting dynamic input and allowed me to test it effectively after everything was made.

# Milestone 3 – Introduction

This Milestone I have expanded the CLI application into a GUI application. Using the Screens developed for the storyboards in Milestone 1 and the SQL procedures in milestone 2, it was a case of stringing the 2 together.

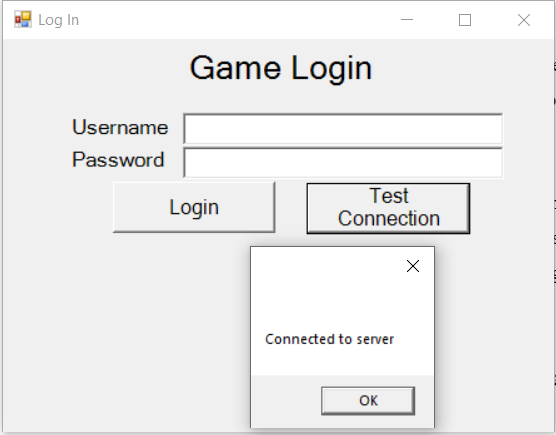
Having made a significant amount of the transactions required for the prototype during milestone 2 and having broken them up logically into different menus in the CLI application, the GUI application followed in the developed and order of the procedures in the SQL. The login screen was developed first, then the character selection screen, game menu and finally the game screen.

Due to the additional requirements of the later milestones, the screens that were initially developed have been modified to include the additional functionality included in the final protype and to meet requirements of the build.

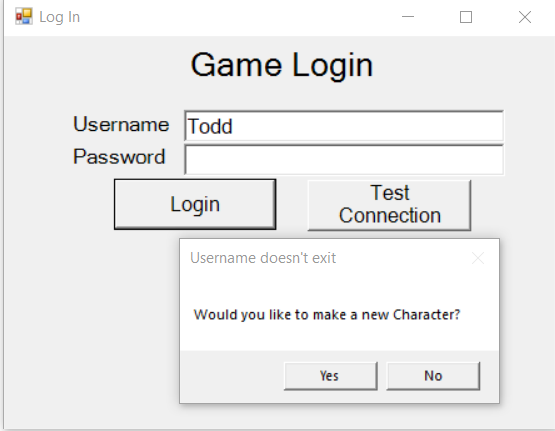
Additionally, minor refactoring of the SQL has occurred which is reflected in the final ER diagram.

Having created the majority of the screen layouts and logic in the SQL, this milestone mainly revolved around working out the logic of how to translate the CLI app to the GUI app.

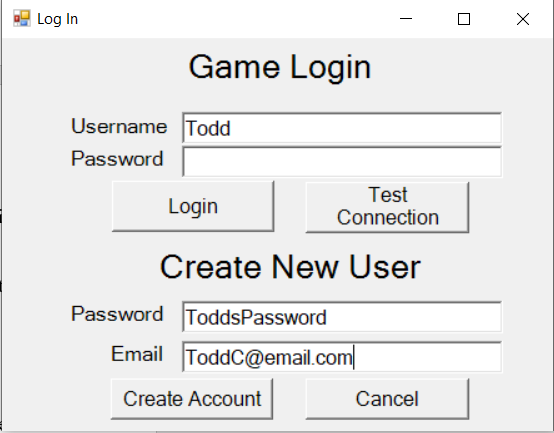
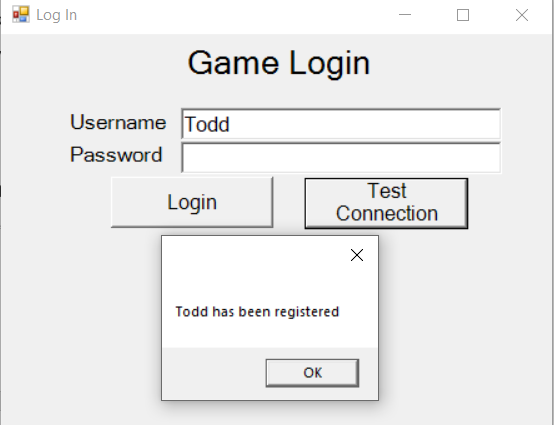
# Storyboards



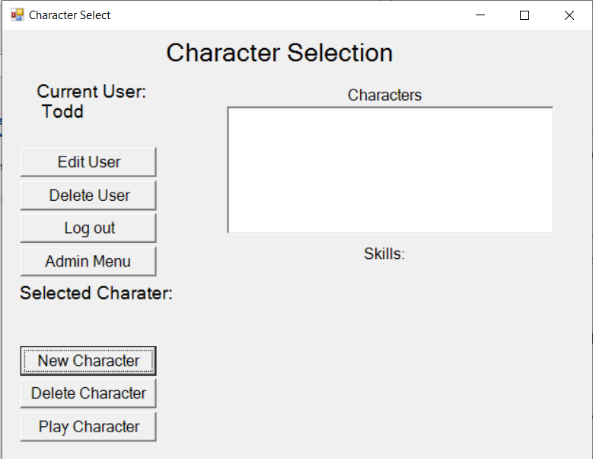
1. The User wants to ensure that they have connection with the database. They click the “test connection button” and are presented with a message box displaying a successful connection message.

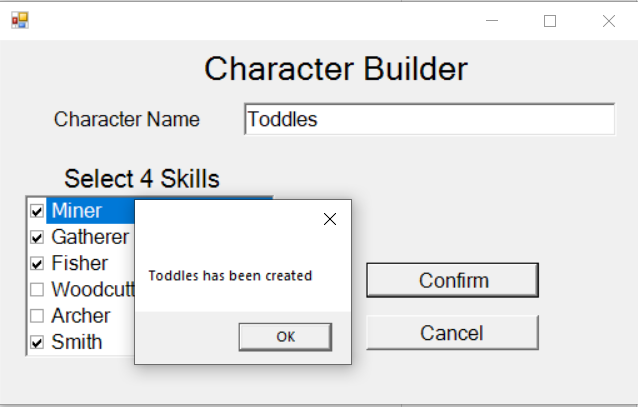
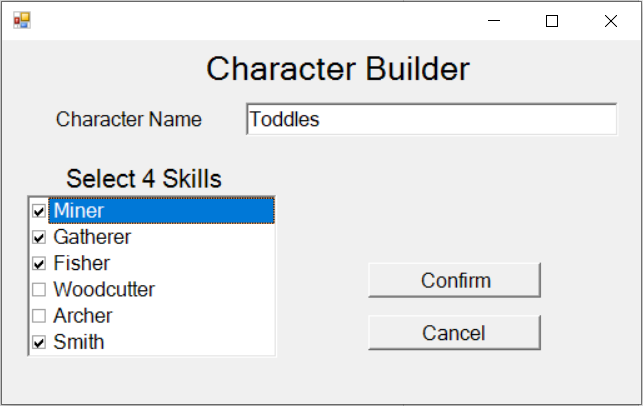


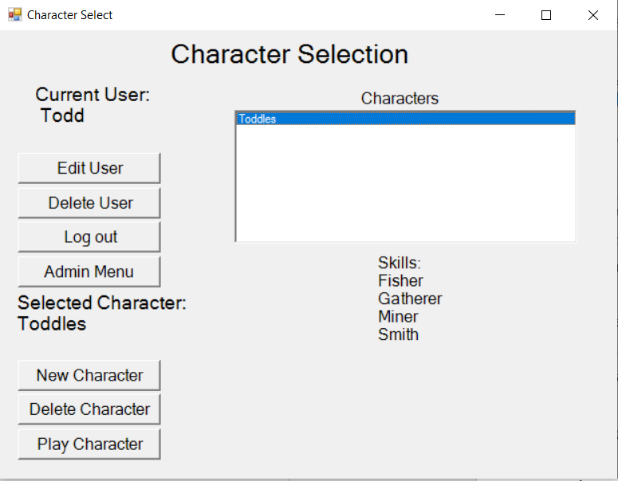
2. The User wants to play the game. They open the application and are presented with a window asking them for their Username and Password. The user enters a username to use in the game as they do not have an account already and clicks login.



3. The user creates an account with the entered username along with a password and email that are associated with the account. If at any time they do not wish to proceed the click cancel to be returned to the previous view else, clicks create account when the entered details are correct.

4. When the user has logged in, they are presented with a character selection screen. Since this user has no characters, they create a new character to play the game with.

  
5. The user creates a character by giving it a name and 4 skills for it to use in the game. When all character information has been added, the user clicks confirm. A message box is displayed with a message from the database.



6. Once the character is made, it appears in a list of characters that the user has made. When the character is selected. The user clicks on the name of the character to select it to play the game with. The list of the characters skills appears.

7. The game menu shows the Characters that are online and ready to be challenged in a game as well as any games the current character is already playing. If the user wants to change characters they can go back to the previous menu with “change character”. Otherwise they select a character from the “Online Characters” and challenges them to a game. If the player is an administrator, then they can access the admin menu from this screen too.

On the right side of the screen is the chat window. Characters can communicate to all other characters by typing in a message in the box and clicking chat.

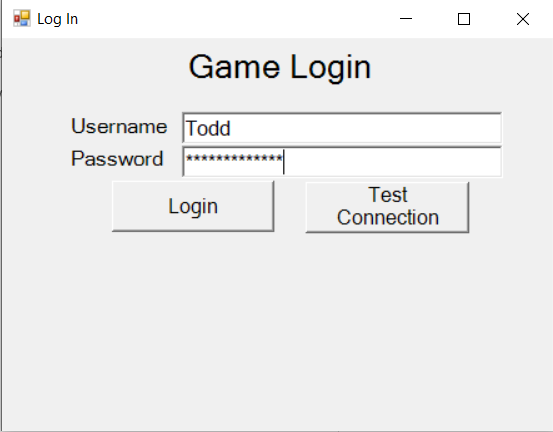
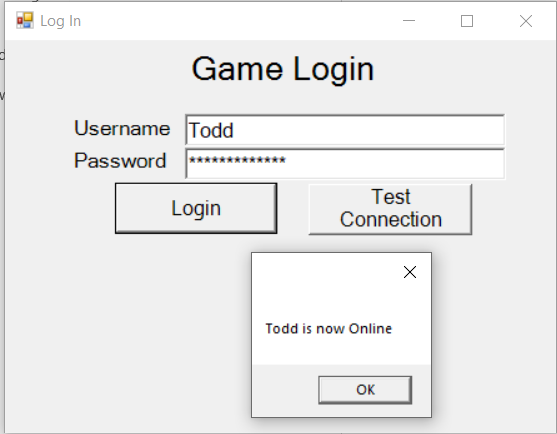
8. When a character has been selected to play against, or when the current character has been challenged to play a game a confirmation box appears to check if the both characters want to play in a game.

9. The Game state starts with both players starting on the ‘home tile’ of the map. Items that need to be collected and ‘mines’ (places to use the items on) are distributed on the map randomly. Characters must first pick up the items that match their skills and use them on the matching ‘mine’.

10. In the above game, Jane has picked up the bow and has used it on the target 2 times earning her 2 points. Mark has picked up the hammer and used it on the anvil 1 time. The score has been updated to show these changes. The items appear in the characters inventory and their quality degrades as they are used. Games procced until an administrator ends the game.

11. A user cannot remember their password to their account. They try 5 different passwords, but after 5 failed attempts they are locked out of their account. Each time he tries to log in after this attempt he is presented with the same “User Locked” message box, even if the password that they entered is correct.

12. To unlock a locked user an administrator must do so from the administrator screen. From the game menu or character selection screen he clicks the ‘Admin’ button which takes him to the admin menu. From here he can kill running games and see a list of the online players, edit characters, edit users and add users. The administrator wants to unlock the locked users account. The administrator unlocks locked account by selecting the name from the list of locked accounts and clicks unlock account. A confirmation message box appears, and the administrator selects yes to confirm. A similar confirmation occurs for all administrator actions.



13. A user wants to log into his newly unlocked account. He enters his username and password and clicks login. A message box is displayed with a response message from the database. On a sucessful login, the user is taken to the character selection screen.

14. A user wants to change their password to something more memorable so that he does not get locked out of his user account again. He clicks edit user on the character selection screen and is presented with the Edit user screen. He changes his password and clicks confirm to record the changes to his account details.

15. A user no longer wants to keep on of their character. On the character selection screen he clicks on the character and selects ‘Delete Character’. A message box asks for confirmation of deleting the character. He selects yes and there character is deleted.

16. A user decides that he no longer wants to play the game at all. He logs into his account and clicks ‘Delete User’. A confirmation pop up appears prompting “Are you sure you want to delete your account?” and he selects yes, and his account is deleted.

17. To re-join a game that a character leaves, from the game menu they select the game that they want to re-join and click ‘Rejoin Game’. A window appears to confirm the choice of the user.

# Finalized ERD

To reflect the changes that have happened from the original draft of the assignment, I have included an updated ER Diagram. It includes all the changes to the database that have occurred and reflects the final physical implementation.

# Finalized Usage Scenarios

\*\*(linked to SQL)

Register a User

Refers to storyboards 2 and 3. New user to the system creates an account.

User Log in

Refers to storyboards 13. A user with an already registered account logs into the system.

Edit User

Refers to storyboard 14 From the character selection screen, a user can choose to modify their account.

Delete User

Refers to storyboard 16. From the character selection screen, a user can select one of of their characters and choose to delete it.

User Locked out

Refers to storyboard 11. If a user has 5 or more failed log in attempts, they are ‘locked out’ of their account and require an administrator to unlock their account.

User Creates Character

Refers to storyboard 5. A user creates a new character that is associated with their user account.

User Deletes Character

Refers to storyboard 15. A user chooses to delete their character associated with their user account.

Select Character to play game

Refers to storyboard 6. The logged in user selects from a list of characters to play the game with.

Create Game

Refers to storyboard 7. Once the user is logged in and has selected character they can choose or be chosen to play a game.

Leave Game

Refers to storyboard 10. During game play a character can leave the game, saving the state of the character in the game and returning to the game menu.

Game Generates Items

Refers to storyboard 9. When the game is generated, the items inside the game are also generated.

Game Generates Mines

Refers to storyboard 9. When the game is generated, the mines inside the game are also generated.

Return to Game in Progress

Refers to storyboard 15. From the game menu when a player has left a game in play they can return to that game

Character makes Move

Refers to storyboard 10. During game play, a character clicks or presses a key associated with a movement to another tile.

Character picks up item

Refers to storyboard 10. During gameplay, if a character is standing on a tile with an item and they take the appropriate action to pick up the item (possibly a key press) it is added to their inventory.

Player Uses Item on Mine

Refers to storyboard 10. During gameplay, if a character has an item that matches a ‘mine’(place to use items) and the matching skill, they can ‘use’ the item (with a keypress). Using the item results in an increase in points and a decrease in the item’s durability.

Character makes a chat

Refers to storyboards 7, 9, 10, and 12. Once a character has been chosen for game play, in gameplay or in the admin panel, a character can chat with all characters.

Access Administrator Interface

Refers to storyboard 12. A character who is an admin can from the game menu access an administration window where they have administration rights over the games, characters and users.

Admin Kills Running Game

Refers to storyboard 12. An administrator can kill any/all running games.

Admin adds a user

Refers to storyboard 12. An administration can add users to the database.

Admin edits user

Refers to storyboard 12. An administrator can modify any user account in the database.

Admin deletes user

Refers to storyboard 12. An administrator can delete any user account in the database

Admin unlocks locked user

Refers to storyboard 12. An administrator can unlock any currently locked user accounts.

# **Implementation of the GUI prototype Application Rational**

## Visual Studio

For the creation of the forms used in the GUI app, I have used Visual Studios form creator. The draft ideas where created in milestone 1, of which I have adjusted for this final milestone.

I have used the NuGet package manager to acquire different packages in the creation of the app. Initially before I received instruction regarding the MySQL.Data package I was using a NuGet package called MySqlConnector which is an open source package that allows .NET application to connect to a range of databases. Fortunately, the code that I had written using the MySqlConnector package directly worked with the MySQL.Data package which provides a library of code for making connections to a MySQL database. I have “rebuilt” the application with both connectors once completed, and both provide the same functionality.

The external calls from the app to the database through the connector have been all isolated to within the data access class. This one class is responsible for retrieving datasets from the MySQL database which the app connects to. While different forms make calls to the class, the forms themselves do not have methods that make calls from the database.

Notably, some of the field types needed to be changed to comply with the MySQL.Data package requirements. TINYINT’s cannot be used with the MySQL.Data package. I have therefore changed all integer types (integer, small integer, tiny integer) to integer to resolve the issues that this made.

Due to making the CLI application mostly return messages, where possible I have converted these into a message box that is displayed to the user. This allows for more dynamic feedback for the user based on the input/s that they provide.

In the cases where multiple data points or a full dataset is returned to the application, I have opted to use list boxes as often as possible. I understand that there was a requirement for using a data grid view, however I have found these to be problematic. Selecting an empty data grid view can trigger null object reference errors and the data grid view lacks some event handers that the list box has such as selected Index Changed. While I am sure that this may have been pedagogical as to work around the issues that data grid views have, I have found that list boxes to be more effective in displaying data, particularly that of the chat messages. As it was listed as a requirement that A data grid view was used, I tried to include it in the game form but after numerous errors with them with no clear way as to how to work with them or any tutelage on working with data grid views specifically in C# including in the resetting of data that is displayed in a data grid view I have opted to not use them in place of the far superior list box. If a list box is not an equivalent of a data grid view it is obviously better. A disagreement to this fact would need to be explained in detail to this student. Notably, the inserting and selecting of items in both data grid views and list boxes is similar.

Visual studio allows for the rapid development of the application and only really the lack of experience of the author slowed down the development of the application. Event handlers can be quickly added to form elements quickly and properties adjusted.

Elements of the code where technically quite difficult for me, and I very often turned to the internet for support. Where I have copied block code from websites I have included a comment as a reference. I’m not entirely sure how to do this to APA v7 standard inside of the code, so I have also included these references in my references of this document. For other parts of code I found either the reference to how the code works from either the Microsoft documentation of C#, w3 schools or stack overflow. In these cases where I have diverged from the reference material I have not included direct references but again have referenced those in this document.

## GitHub

To ensure version control, I began using GitHub halfway through Milestone 2. Not only did this mean that I had all the files for the project in the same repository, but I had a backup of all the work that I had done in the project to that point. While I have not needed to use the backup, I found this significantly better in terms of assurance that my documentation, that of the word files, where backed up. I have had significant issues with one drive in the past and converting to GitHub simplified file management significantly, as well as ensuring that after committing and pushing local changes, I had a definite backup offsite.

## Visual Studio Code

For writing the SQL, I found the MySQL workbench to be difficult to work with. Therefore, I opted to use Visual Studio Code as it primary has a dark theme which was easier on the eyes at night.

I downloaded extensions to aid in writing of the SQL code. Of particular use was the SQLTools extension that allowed me to make connections to the database and the drivers extension for the MySQL database engine specifically. These together allowed me to make connections to the database and to execute queries directly from Visual Studio.

While working on the procedures I would often find that I needed to look back at the structure of the tables that I had created, so for this I simply split the editor and scrolled up to my DDL in one pane while continuing to write what ever procedure I was working on at that time in the other pane.

As with MySQL workbench, visual studio code has syntax highlighting that aids in the creation of code. With the .SQL file type, the language was automatically set to SQL in the editor and aided in the quick writing of the code.

While not that much different from MySQL workbench, I found that some of the keyboard shortcuts and tools available in Visual studio code such as find and replace to be significantly helpful in producing the SQL. While I could have technically coded entirely in MySQL workbench, I found the modularity and overall style of visual studio code to work for me better.

## MySQL Workbench

While visual studio code was the main editor of choice, whenever I ran into issues (occasionally the connection would not work between visual studio code and the database) or I couldn’t resolve a bug I reverted to the MySQL workbench. It provided a more detailed output window that allowed me to trouble shoot the error that I was running into by providing either a line reference or an error code.

Often as well, visual studio would not catch small syntax errors that MySQL workbench would. The combination of the 2 text editors made for quick debugging. It might have been possible to have found an extension in visual studio to do what I was after, but it was also useful to use an entirely different editor to double check my work. Both editors had different tools that where useful at different times. I would not say that either was better than the other, just that they do/did different things. I can see that with the modularity of Visual Studio Code with its extensions that it could easily surpass that of the MySQL workbench in time, however MySQL workbench is specifically designed to write code for MySQL and likely will be superior for that reason.

# Milestone 3 - Conclusion

# References

MySQL 8.0 Reference Manual :: 15.1.1 Benefits of Using InnoDB Tables. (n.d.). Retrieved June 1, 2020, from https://dev.mysql.com/doc/refman/8.0/en/innodb-benefits.html

MySQL 8.0 Reference Manual :: 15.2 InnoDB and the ACID Model. (n.d.). Retrieved June 03, 2020, from https://dev.mysql.com/doc/refman/8.0/en/mysql-acid.html

Ian. (2017, October 08). What does ACID mean in Database Systems? Retrieved May, 2020, from https://database.guide/what-is-acid-in-databases/

MySql.Data 8.0.20. (n.d.). Retrieved June, 2020, from https://www.nuget.org/packages/MySql.Data

The report is assessed by considering the following:

a. Sketches of interactions in storyboards. These must cover all interactions

described in the project brief, and must be accompanied by written

narrative that describes the interactions. [2]

\*\*Update to the latest storyboards screen shots and narrative\*\*

b. The report includes descriptions of the design of usage scenarios of the

game. [2]

\*\*Update and include new usage scenarios\*\*

c. The report describes all use cases from your storyboards. [2]

\*\*Update??\*\*

d. The report refers to SQL that has been refined, developed, and refactored to

meet use cases. [2]

\*\*Refer to sql\*\*

e. The report includes a written description of how you implemented your

game’s GUI prototype.[2]

(10 marks)

\*\*Write on implementation\*\*

2. Extend your work in Visual Studio (CLR).net to include a series of Windows Forms.

There is to be one for each screen or layout in your game. The forms are to work,

passing data from the fields to event handlers. You are to implement at least:

○ the log in, [2]

○ choosing from a list of online and available players, [2]

○ game confirmation, [2]

○ a game display, [2]

○ an administrator’s window allowing for player detail display. [2]

(10 marks)

3. Create a class that has methods for each of your previously defined SQL database

procedures. This class is to provide all functionality implemented in the database.

○ Each method is to be activated by some action relevant to your

game, even if it is a simple test button that causes the action. [5]

○ You are to check user login by calling a method in this routine. [3]

○ You are to produce a list of online and available players and display

them in a DATAGRID (or equivalent). [4]

○ You are to implement selection from the DATA GRID.[3] \*\* Explain in detail about the selection of a game from a data grid. Be explicit as to why you have not done this in all other areas.

(15 marks)

4. Implement one genuine interaction in the prototype that causes a player to move in

your game. (5 marks).

Marks are given for describing your rationale and processes, as well as plain descriptions of

the database and application you are developing.