|  |
| --- |
| DAT602 Final Report |
|  |

|  |  |
| --- | --- |
| ID: 13508189 | Celeste Quinn |

DAT602 Milestones 1, 2 & 3 – Final Report.

DAT602 Final Report

Celeste Quinn

Table of Contents

[Milestone 1 2](#_Toc75776394)

[Game description 2](#_Toc75776395)

[Idea 2](#_Toc75776396)

[Gameplay 2](#_Toc75776397)

[Players 3](#_Toc75776398)

[NPCs 3](#_Toc75776399)

[Items 3](#_Toc75776400)

[Shops 3](#_Toc75776401)

[Storyboards & Usage Scenarios 4](#_Toc75776402)

[Main screen 4](#_Toc75776403)

[Create Player 5](#_Toc75776404)

[Login 6](#_Toc75776405)

[Locked Account 7](#_Toc75776406)

[Character Management 8](#_Toc75776407)

[Admin Management 9](#_Toc75776408)

[Edit Account 10](#_Toc75776409)

[Gameplay 11](#_Toc75776410)

[CRUD table 12](#_Toc75776411)

[Crud Analysis 12](#_Toc75776412)

[Entity Relationship Diagram 13](#_Toc75776413)

[DDL 14](#_Toc75776414)

[Test Queries 14](#_Toc75776415)

[Milestone 2 14](#_Toc75776416)

[Game Activities 14](#_Toc75776417)

[Activity Identification 14](#_Toc75776418)

[Character Management 14](#_Toc75776419)

[Admin 15](#_Toc75776420)

[Movement 15](#_Toc75776421)

[Battle 15](#_Toc75776422)

[Interact 15](#_Toc75776423)

[Inventory Management 15](#_Toc75776424)

[Chat 16](#_Toc75776425)

[Logout 16](#_Toc75776426)

[Activity Procedures 16](#_Toc75776427)

[Multiplayer Gameplay 16](#_Toc75776428)

[Test Data 18](#_Toc75776429)

[Changes 19](#_Toc75776430)

[References 19](#_Toc75776431)

[Milestone 3 20](#_Toc75776432)

[Adjustments 20](#_Toc75776433)

[Updated ERD 21](#_Toc75776434)

[Storyboard Interactions 22](#_Toc75776435)

[Moving 22](#_Toc75776436)

[Implementing Gui 31](#_Toc75776437)

# Milestone 1

## Game description

### Idea

This will be an RPG game where a player starts off randomly in the world. The map the game will use will be a 100x100 grid of tiles. The player may move from tile to tile by expending stamina or an action point system of some kind, and also perform actions by using the same resource. On the tiles, there may be an item, an NPC, another player, or a shop. Or any combination of those objects. The goal is to enhance your own player character to become stronger, and take on other player characters.

### Gameplay

To move from square to square, players expend stamina, which recharges on a timer. There are items on each tile they can pick up and add to their inventory. The item could be currency, a weapon, or armour. Once a player picks up and item, it is removed from the map. Currency thinking of a system where there is a set amount of items always in game at any given time, or that once an item has been picked up, it respawns on another tile after a duration, to keep it equal and prevent players from farming everything. So in keeping with the game’s requirements, a tile cannot be occupied by more than one player at once, therefore in order to have a PVP based system, players must be able to interact with other players by being adjacent to their tile. There would be commands that pop up when in range of another player. A basic ‘Attack’ at first, and as the player progresses and unlocks more skills, they have more options they can use. Players may also encounter NPCs on a tile, which they can attack and if the NPC is killed, will reward them with currency. The use of currency is to accumulate and spend in shops, which again are situated on random tiles, that the player can interact with and buy gear upgrades and also different skills. There would be multiple shops in the game that sell different things.

### Players

Each player starts out on a random square. Players can encounter other players and attack them while on adjacent tiles. When one has reduced the others health to 0, they get a percentage of the defeated players coins. Once a fight has been engaged it would lock both players in battle until it was finished, to prevent complications of what might happen if either tried to move or run away. When a player is defeated I’m thinking they would either be sent back to a home tile which would be in a safe zone, or the player becomes protected by a timed aura that makes them invulnerable to any more attacks. This would depend on how difficult these options are to program.

Players have an inventory which holds their gear and any items they have. They also would have stats which would be affected by their gears upgrades. At the moment I have 3 primary stats in mind- Attack, Health, Stamina. Attack would be the strength of their attacks, Health is obviously their hitpoints, and Stamina is used for how many actions/steps they can take in the game. When the player’s health is not full, say that a fight reduced them to 1/10 health, it slowly regenerates according to a timer (or possibly 1 HP per action taken? Like moving from tile to tile could regen HP as well).

### NPCs

These non-player characters are scattered across the map as well. A player can arrive on a tile occupied by an NPC and initiate an attack on it. If they win then the NPC will drop coins and despawn. Depending on what system of map I end up going with, these NPCs could have different levels, the lower level ones would spawn only in lower areas, and higher ones in higher areas – or they could all scale with the player’s level. There is also the possibility of having boss level NPCs that would drop unique items that give players extra skills.

### Items

Gathering these is one of the main gameplay elements. Items can come in various forms. There would be 3 regular forms of items, which would be randomly generated- out of attack items, hp items, and stam items. Think of them as a powerup to the player’s stats. So when collected they provide a permanent +1 bonus to the relevant stat.

### Shops

This would be a tile with a vendor on it, that the player can buy items/new skills for a set price of coins, which they will have been accumulating from playing the game and killing monsters/other players. The item or skill they buy then goes to the players inventory.

## 

## Storyboards & Usage Scenarios

### Main screen

Diagram

Description automatically generated

#### Diagram 1

* 1. New player wants to create an account, clicks this button
  2. Returning player wants to login

This is the main screen everyone sees upon first opening the game. They can then decide to make a new character or login to an existing account.

### Create Player

Graphical user interface

Description automatically generated

#### Diagram 2

2.1 Enter username to register with  
2.2 Enter password to use for login  
2.3 Back button takes you back to the main screen  
2.4 Confirm creation of new user with specified username and password

The new player registration screen. Players would enter the username and password they want then press create button. If the name has been taken already, the message above 2.1 appears and they must re-enter a unique username.

### Login

Graphical user interface, application

Description automatically generated

#### Diagram 3

3.1 Error message if incorrect user details have been entered  
3.2 When valid credentials are inputted then take to logged in screen

Already registered users will enter their details at this standard login screen. The 3.1 error messages appears when they input the wrong password associated with the username they have entered. A single username can take up to 5 invalid logins before the account gets locked and the player must take action to unlock the account, try a different username, or go back and create a new account.

### Locked Account

Graphical user interface, text

Description automatically generated

#### Diagram 4

A popup dialogue to inform the player to check their emails for further instructions. They would receive something like a password reset or maybe the password could be sent to them in plain text like the good old days. Even possibly just replying to the email making sure it was them who tried to access the account and for the admin to unlock the status of their account.

### Character Management

Graphical user interface

Description automatically generated

#### Diagram 5

5.1 Displays logged in player’s username  
5.2 Lists character associated with registered account  
5.3 Selected action  
5.4 Executes selected action

This is the next screen that appears after the player has successfully logged into their account. The back button here logs the user out and takes them back to the main screen. I thought the user also should have control over whether they can delete their own character or not.

### Admin Management

Graphical user interface

Description automatically generated

#### Diagram 6

6.1 Lists every account that has been created  
6.2 Admin-specific commands that can be executed

Admin accounts can update player details, log out any players from the game that are currently logged in, create new players, and delete players. Admin must confirm before executing any of the actions. The back arrow on this screen functions the same as in Diagram 5, logs the admin out and takes you back to main screen.

### Edit Account

Graphical user interface, application

Description automatically generated

#### Diagram 7

7.1 Text fields with account information  
7.2 Checkboxes to display whether account status is ‘locked’ or ‘admin’

For updating user details. Text fields are editable and updated information is saved upon pressing the save button. The cancel button exits the screen and disregards any unsaved changes. The locked checkbox can be unticked here if an account has been locked, to make it accessible again. The admin checkbox can be ticked to change a player’s status to have admin rights, which just made me realize that admins still need to have the play action in their character management screen. Will adjust that screen on the wireframe program I’m using.

### Gameplay

Graphical user interface, application, map

Description automatically generated

#### Diagram 8

8.1 The game screen. Player avatar moves around here and plays the game  
8.2 Action buttons. Attack and inventory. Attack is for when you encounter NPCs or other players, it will initiate a battle. Inventory stores players objects and skills.  
8.3 Directional buttons, used for movement across the map  
8.4 Player stats and currency  
8.5 Global chat box. Could also serve as an events log to display things such as battle outcomes  
8.6 Input box for game chat

This was originally just a green and blue filled box but I wanted to long for some cute graphics that’ll probs never end up in the final version. Also created that character sprite specially on some sprite design website. After making this I thought about it and had some more ideas for the UI but this will have to do as it’s a pain to re-edit it right now. This design is mostly just wishful thinking as well as I don’t exactly know what the final project will look like and how much we could even do, as well as the restrictions of programming in SQL. It may even just end up as a white screen with no graphics at all except player sprites and NPCs. I don’t even know if its possible to do something like this at all without a proper game building program like Unity or something either.

## CRUD table

Diagram

Description automatically generated

### Crud Analysis

This is an expanded example table that I filled in. I could not remember what we covered in class about CRUD and this is the only resource that was on the course page. So not really sure what other events I can add at this time. Covered every event that I could think of.

## Entity Relationship Diagram

Diagram

Description automatically generated

Here is the logical ERD for the database I have so far. I was a little confused about linking up the stats to the character but thought I could do so by just having the character ID in the stats table thereby linking it that way? Or would the stats table even be needed at all and the stats should be as columns in every individual character? That is something that may have to be worked out as I implement actual data to see how it should work. The notations on visual paradigm did not behave as I would expect for one of the relationships- itemID linking to character. I wanted it to have that one character could have many items. But the program would not allow me to create a many notation on the side that pointed to itemID in item table, only let me put many in character table. So that led to me being more confused about what side means what with the notations. We covered this a lot in DAT502 but I still get confused when I think about it. So hopefully this is correct at the moment and will probably need to be updated and expanded upon in later revisions.

## DDL

Uploaded to GitHub repository [located here](https://github.com/d3aths/DAT602-project/blob/main/DAT602%20Celeste%20Quinn%20Milestone%201/ddl%20scripts.sql)

## Test Queries

Uploaded to GitHub repository [located here](https://github.com/d3aths/DAT602-project/blob/main/DAT602%20Celeste%20Quinn%20Milestone%201/test%20queries.sql)

# Milestone 2

## Game Activities

### Activity Identification

#### Register and Login

The first activity a player needs to do to access anything else in the game. Register includes signing up with a username and password. This creates an account that can be logged into. Logging in requires the user to enter a valid username and password.

#### Character Management

Once logged in the player is presented with the options as shown in Diagram 5 of M1 Report – A list of the character names and options to play and delete. This is how they select their player and confirm entering the game.  
Upon review I feel another option needs to be added here – edit. Which presents you with the edit player options that only the admin could previously access (D6 M1), so the player can update their own account details and add an email address, but obviously locking the account and isAdmin checkboxes would be disabled for regular players.

#### Admin

Admin functionality includes the ability to edit player’s information, lock accounts, make them admins, kill a session, create a new player, and delete a player.

#### 

#### Movement

When the player is in the game their player character must be able to move in either North, East, South, West. Movement is restricted to the squares directly around the tile they are currently on. Movement requires the use of one point of stamina, a character stat that can be increased. Depending on how difficult this is to implement, this may be removed by the final project.

#### Battle

If adjacent to a tile with another player or NPC on it, the player can initiate a fight where their stats are pitted against the opponent to find a winner. This is how they gain coins and sometimes items. I am as of yet undecided about a levelling system. My original plan was to use the items to adjust the players stats, therefore the better items the player had, the higher their stats were, but there’s also the classic experience system where a battle would reward experience and the player gains higher stats each level they gain. It will come down to implementation to fully decide on this.

#### Interact

A player can find items on the map randomly and possibly also interact with shop objects that appear randomly on the map and can use their coins to purchase items from. On paper this seems easy enough to implement, but also not sure yet. So the only solid interaction the game has would be collecting items off the ground as they move. So there needs to be a command for this, and a GUI prompt to show the player they are able to interact with something nearby.

#### Inventory Management

Clicking the inventory button from the main GUI should open a frame that shows what items the player has obtained, including their descriptions and value, and if applicable what actions they can take with the items.

#### Chat

Pressing the enter key (universal standard) or perhaps clicking on the input area on the chat box, would allow the player to talk to anyone else in the game (enter to send the message as well) If this were to be developed further there would have to be some options of reporting or ignoring a player you do not want to hear, but that is beyond the scope of this project.

#### Logout

Ends the session and saves the state of the game so the player can login next time and return to exact how they left off.

### Activity Procedures

Stored procedures are saved in the SQL file ‘database creation and procedures’ attached.

Player registration procedure: addCheckedUser (not used in visual studio currently), AddUser  
Player selection: pickClass  
Live gameplay: move  
Confirmation for a game/show online players: showPlayers  
Game admin functions: editAcc, delAcc

## Multiplayer Gameplay

Multiplayer gameplay will be supported because all of our interaction transactions are following the ACID protocol when they are executed. A transaction is a collection of procedures or sql statements that work together as one and is treated as one whole transaction. The acidity of transactions make them perfect for use with multiplayer games as multiple functions working as a single operation is ideal for how reliable it needs to be.

Lets take a couple of transactions as an example to use while relating them to being acidic, a transaction for a fight, and a transaction for moving.

Firstly, they have to be **Atomic.** This means that since everything inside the transaction works as one, everything has to either be all successful or fail completely. This means that if a transaction had 10 different statements inside it, and 9 succeeded but one failed, the whole thing would fail. The transaction is not allowed to be only partially working. In terms of our fight transaction here, it would hold all the statements necessary to work out the outcome of a battle. It would compare the players stats with the opponents stats, calculate damage based on that, report the outcome to any players involved and update their games accordingly. Not a single step in this is allowed failure, it must follow that flow or it will fail entirely – like we can’t just leave it in mid fight.

Secondly, the transactions must be **Consistent**. Which is to say that the data must obey the rules and stay in a consistent and valid state. One example for this is that in some games, a player might try to use hacks to change their stats or gold level (mostly only achievable in client-sided games) So consistency here would prevent someone from going in and changing their stats as that would not reflect on the records the database has, because if the data is not consistent, any changes will be rolled back and data integrity is maintained. Another is that gold value cannot be negative, only positive, so if a player was trying to buy an item from the store, they cannot buy something that is over the value of gold they have on hand, as it would bring their balance into the negatives. If this is attempted, the transaction is cancelled. The transactions the game have must account for situations like this.

Next, **Isolation**. This seems to have a few different meanings depending on which website is explaining it. One is that the transactions are not able to be interfered with by other transactions while they are busy doing their thing, or in an intermediate state. The transactions are not dependent on one another, so their isolation means that you wouldn’t need one transaction’s success to be reliant on another transaction that was happening at the same time, to also succeed beforehand, and also that the outcome of multiple transactions happening at the same time, will not affect their ability to succeed. The moving transaction would be a good use case to relate to isolation, as multiple players will be moving around the map at the same time, the transactions must ensure that no one is impacted by other’s movements, that everyone can successfully move around, asides from the rule that you can’t be on the same time as another player. The SQL must meet this rule.

Finally our transactions must have **Durability.** Which means that after a transaction has completed. The changes are permanent and will not be reverted if anything happens to the server/system/database. This is one of the most important things in gaming, especially online gaming, as server outages or disconnecting is one of the most annoying problems. Having a rollback of your progress because of a crash is every gamers worst nightmare, especially if something happens like, you’ve just killed a really hard boss, or you’ve received an ultra rare drop. The last thing you want is that being taken away from you. So for the transactions to have durability, they need to be not able to be undone.

At this point in the Assignment there are no transactions to ensure this, however there is some basic groundwork so far that could count as multiplayer support. This is in the form of the first iteration of the move() procedure. If the player attempts to move onto the coordinates that is occupied by a user already – there will be an error message displayed and they will be unable to complete the move.

## Test Data

C# work will be shown in the attached file “datconnection” solution.

In the visual studio project, the test data is inside the Program class, and dataAccess class. These will not be renamed as they were creating during a class session using the names we used then for testing, and I kept going on these files. I have learned the hard way that renaming things in visual studio breaks your whole solution because the process of renaming it doesn’t change everywhere that the file is already stored in other parts of the solution and I’ve already lost work attempting this for other activities.

Anyway the methods that call the procedures from mysql are in DataAccess, and then you can find where to input information in Program, where you can change the variables. As long as you follow the conventions for input as defined in the comments, there should be no errors. It comes with predefined data so you can just build and start straight away, and press enter in the console window that appears to continue to the next function after one has completed, it will give dialog queues. You can then check the output in mysql workbench to ensure the data has been written successfully to the database.

The procedures that have been written for this milestone, should cover the basic functions that the game needs to have in this early development stage. Registering a user account, being able to edit details, picking a class for the user, and then moving- as well as the extra procedures like showing players and deleting accounts. With no interface it can be tricky to visualise how these will work with the game in the end, and by this I mostly mean something like login validation. After taking web last year it seems to me this might have to be done with something like an outside authentication server like firebase? I don’t know. Maybe we have covered it in class and I’ll have to go back through and look to see if it can be done simply in mysql or not. It’s hard to remember everything if you’re not up to the point where you’re actually putting it in the development of your game actively.

Further procedures will have to be developed in future to include more of the activities as described in the above section in this report.

## Changes

A list of changes and reasoning from M1 to M2

* Added charID FK into table ‘account’  
  The account needs to know what characters have been created on it. Not sure if this also means that userID key should be removed. Currently userID is account’s PK and is an FK on character as well, so it shows what userID the character is associated with. This is somewhat confusing to explain – so – there is one account that belongs to the user that has the ability to have multiple characters on it. The account has a userID. And each character also has a characterID. Each character has a reference to the account they were created on via the userID, so each account has a list of charID’s that are on it. So unsure if both are really necessary.
* Added new column ‘status’ into character table  
  This is so we can complete the required query for confirmation of game, instead this will show the currently online players. So characters need to have a status showing them online or offline, and will be updated accordingly when a character joins the game.
* Deleted character table entirely and linked up the old rows that were still needed into account table. This was annoying because having an account and trying to link up the characters was complicated. It worked to an extent, I could link a new character to the last account that was created, but as there is currently no way to log in to an account and then be in it while creating a character to link it like that or something, like I just have no idea how that would have worked. This pretty much deprecates the above changes I’ve made but will still leave them in here for posterity. It also means I won’t have a procedure for the criteria of Player selection, as that was the old createChar() procedure which I’ve removed, so have to think of something else for that I guess.
* Added a template table that new characters must choose from when first starting, that determines their stats. This takes the form of 3 different builds a player can choose from. They are Berserker, which has an increased strength, Tank which has increased hitpoints, and Ninja which has increased stamina.
* Deleted map table as it was not being used

## References

Watts, S. (n.d.). *ACID Explained: Atomic, Consistent, Isolated & Durable*. BMC Blogs. Retrieved May 18, 2021, from <https://www.bmc.com/blogs/acid-atomic-consistent-isolated-durable/>

ACID versus BASE Data Stores. (n.d.). *Dummies*. Retrieved May 18, 2021, from <https://www.dummies.com/programming/big-data/hadoop/acid-versus-base-data-stores/>

Roe, C. (2012, March 1). ACID vs. BASE: The Shifting pH of Database Transaction Processing. *DATAVERSITY*. <https://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/>

*IBM Docs*. (2021, March 29). <https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions>

# Milestone 3

## Adjustments

As at the time of writing this report, I have not received any feedback for Milestone 2 yet so I will have to describe what I have changed based on feedback for M1 only.

I have adjusted the Entity Relationship Diagram several times over in various reworkings of the design of my database based on functional requirements when developing the SQL procedures further.

My SQL is all indented within procedures, and every query is also within a procedure.

### Updated ERD

Diagram

Description automatically generated

## Storyboard Interactions

### Moving

Graphical user interface

Description automatically generated

Players move around a map from one tile to another, collecting items, competing with other players who are playing on the same tiled map.

This use case is demonstrated in the game window. The storyboard has changed from M1 to reflect how it actually looks within visual studio. The map tiles would change based on movement that was done by clicking the directional arrows, and attacking other players/npcs as described in the original game description would have been done by enabling the attack button while adjacent to a tile with something else on it. The tiles would be visually updated to show if there were items on it, and maybe some kind of markings within the game window to give the illusion there was a bigger grid behind what was displayed on the window, that the player was traversing.

The SQL command I wrote for Move enables this to happen via calling the procedure every time the player moves, it updates the X and Y coordinates assigned to the user, in the database. The move can only successfully complete if the coordinates are not already assigned to another user within the database, thereby preventing a player from being on the same location as another player, which was a requirement for the game in the brief.

In C#, this happens by assigning variables to each box you see in the game window grid, with the boxes that are adjacent to the current box in all 4 directions. Clicking one of the movement buttons would look for what box was assigned to say, the tile south of current location, and move the player there via updating the image on the box, to give the effect that you are moving, and then assigning the newly moved to box as current tile. Each move writes new coordinates to the database by calling the Move procedure.

Graphical user interface, application, website

Description automatically generated

A player is logged in and registered

Creating a user is as simple as opening the login screen after clicking the enter game button on the main page. Clicking on create user after entering a new username and password will create a new player object and assign that to the current player. It checks the entered values with a function I wrote in C# the accesses the MySql data via a reader, and compares it with the username column in account. If it finds a matching username it will reject the new user creation, otherwise, it will call on the addCheckedUser procedure from MySql and add the new user to the database. This can be verified in Sql Workbench program after successful user creation, and then you are able to login using those details. Logging in also creates a new player object and assigns you as the current player, and calls the procedure CheckLogin from Sql to match an existing username in the database to verify that the user you are trying to enter exists.

Graphical user interface, application

Description automatically generated

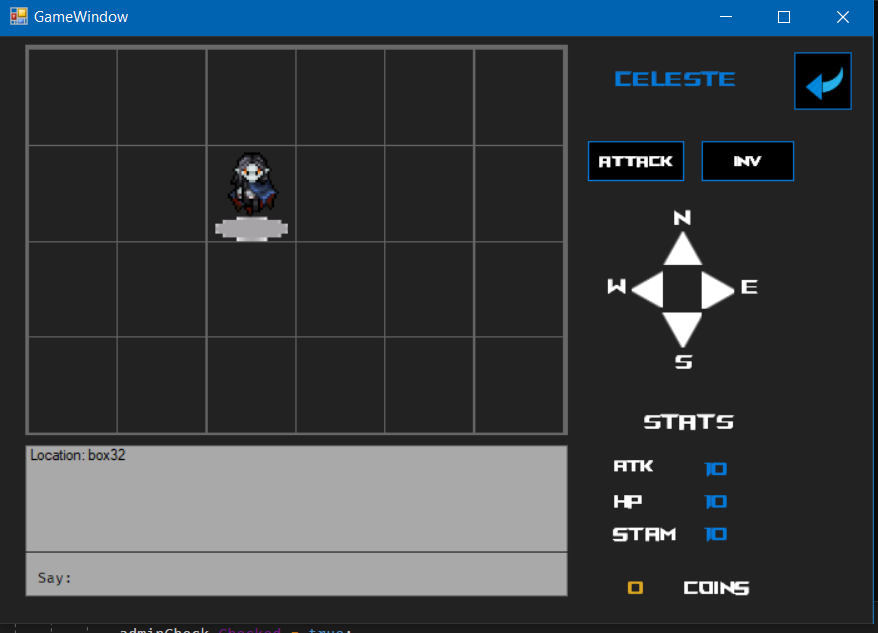
An account is locked after 5 unsuccessful login attempts

Inputting the incorrect password 5 times locks the account via the CheckLogin procedure that is called when clicking the login button. It adds attempts to a variable and after 5 you get the message below that your account is logged out.

Graphical user interface, application

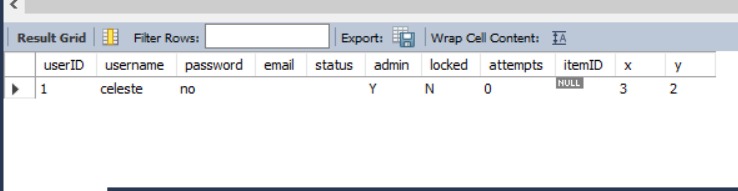
Description automatically generated

Locked out window view, after this screenshot was taken I edited the textbox to displays an administrator email as per the requirements.

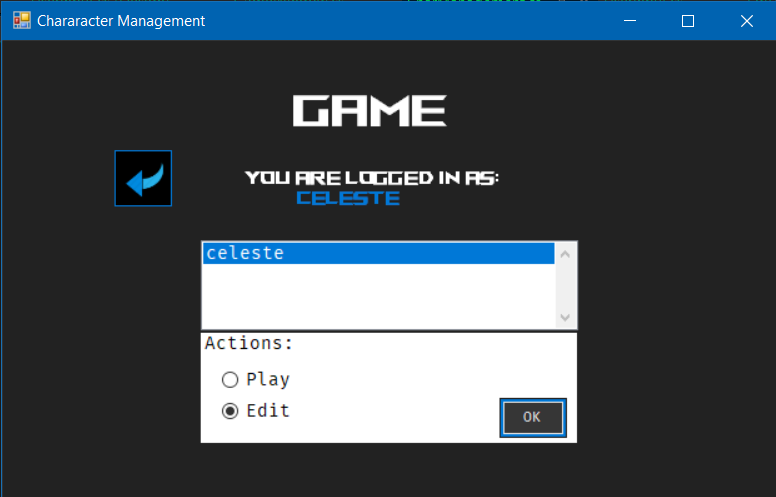


A user starts on a home tile and the tile becomes empty when they move. Players move from one tile to another by clicking. As the player moves, the location is stored in the database

The movement described in the above section detailing the Move procedure. The player starts on the top left tile and there is a variable called lasttile that gets assigned after the database location has been written. The way it should work was that on form load, it gets whatever was in the last tile and sets the current tile, therefore player location, as the last known tile. However this procedure is not working as the way I understand it, every tile would have had to have been actual objects instead of just pictureboxes. I spent hours trying to figure a way to get this to work around it, but was unsuccessful. So the player’s location is still stored in the database as you can see from the screenshot above, the player is on box 32, which has the coordinates of 3, 2.



This is the result from the account table after having moved to that location, showing that the location coordinates are stored.



Non admin character management

Graphical user interface, application

Description automatically generated

Admin character management. Administrators can delete accounts from here which is just used by removing the player object from the locally stored list and then calling on the DelAcc procedure from MySql to remove it from the database at the same time. The way I have done this is that the delete button only removes whoever is assigned to the current player, so the currently logged in player. This is because I could not get the player list on the character management screen to successfully link up to the list of users that the datareader initially pulled in from the database. I tried for a couple of days to populate the list and wrote various functions trying to perform this, but they all failed. The closest I could come was actually pulling all the info into a list, but it would not display as usernames in the list, it only said (Collection). I left the code in that I tried for posterity. This would have been the PopulateList() function in the class for that page.

Graphical user interface

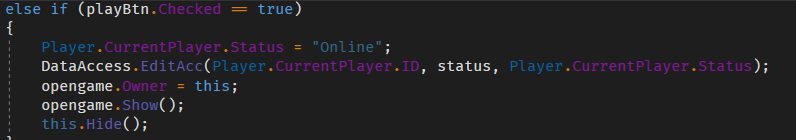
Description automatically generated

Editing any field in the edit window that pops up calls on the EditAcc procedure from MySql. It takes the inputs from the text fields and uses the procedure to then update the information in the account table within the database.

Graphical user interface, text, application

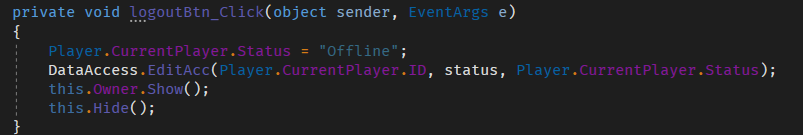
Description automatically generated

Edited data being saved.



When a player leaves the game, their current status is saved in the database.

The above screenshot is the code that opens the game window for the current player. When clicked, the status is set to online and then the EditAcc procedure is called on which then sets the status to the database.



This is matched with the logout button in the game window, where the status is then changed to offline and written again to the database.

A screenshot of a computer

Description automatically generated with medium confidence

Communicating via the chat box is done via a string builder that appends the textbox to the above label to show messages.

Player selection criteria was replaced with the user being able to pick their class, as described in the M2 documentation. This gives them an option to pick from a template with predefined stats that gets loaded onto their character. This works by checking if the current player has a strength value of less than 20, then that shows the player has not been into the game yet and picked their class, thus prompting a choice to pick the class via the stored procedure PickClass. After that, stats are stored in the database assigned to the current player’s ID, and shown in the game window.

## Implementing Gui

At first the GUI was designed based off the original prototype that was made in Adobe XD. The windows forms were recreating each scene from there along with changes made according to development needs. Some certain things were not aesthetically matching to what was originally in the prototype but overall remained very similar to what was planned.

The code for the functions was very difficult to get running up at first but once I had mastered the use of datareader it became slightly easier, except for in some cases that I have detailed in the description of use cases above.

Obviously from M1, everything I had envisioned for this game would have been far, far more complicated to actually implement that I could have done in the time I had. As it stands there is quite a bit of missing criteria from what is needed, but I have stayed up until 3 and 4am for the past 2 or 3 weeks almost every night, working on every assignment I had, and it was still not enough time to be able to finish every requirement for this one. Nevertheless, I am happy with the amount of work I was able to bust out in the last few days as it was a massive amount of coding to pull from nowhere, although very unsatisfied with not being able to fulfil every requirement. There is no instance of the game that can be used for multiplayer gameplay for players connecting to the same instance and being able to view each other’s movements across the game grid, I had no idea how to implement that. There is one genuine interaction that causes the player to move as described in the above sections.