DAT602

Brayden Dunn

ID: 13513282

Assessment one, stage one.

# Table of contents

[Table of contents 1](#_Toc367359488)

[Game description: 1](#_Toc934394330)

[Requirements: 1](#_Toc2124259140)

[Difficulty scaling: 2](#_Toc945608159)

[Logging in: 2](#_Toc416968896)

[Menu Screen: 2](#_Toc1835602595)

[Options: 2](#_Toc934222451)

[Game lobby/Play: 2](#_Toc222676339)

[Gameplay: 2](#_Toc1989495021)

[Administration: 3](#_Toc1906592599)

[Storyboarding: 3](#_Toc1887571827)

[Entity relationship diagram 11](#_Toc1936405687)

[Entity relationship diagram rationale 11](#_Toc567630996)

[Crud Table: 13](#_Toc1231727033)

[DDL 13](#_Toc1616384979)

[References: 13](#_Toc265802283)

# Game description:

The game idea is a 2D based ‘Dungeon crawler’. The player will explore a generated map and encounter enemies with the ability to fight or run away from encounters. The game won’t have an ‘end’. However, with each ‘level’ the player clears the dungeon will become harder and the game finishes once the player’s character dies. The scoreboard system will work off the fact that the player will have an associated level with the deepest floor they reached; this will be used to compare scores against players I.e. Player34 : Level 13 : floor reached 4 : . The game will not have a maximum time limit and the user may save/log out at any point. The game is required to be multiplayer, and a concurrent player may traverse the dungeon in a CO-OP scenario, where the final score is shared among the two players.

# Requirements:

Game must have a GUI (graphical user interface) for the user to interact with and the game is required to be multiplayer.

The requirements devised for the game are as follows:

* Player login, and registration
* Player lockout upon ‘x’ amount of failed sign-in's
* Player may delete their account
* Game tiles are generated
* Items are placed upon those tiles
* Players can move on generated tiles.
* Score system for player
* Player inventory
* Npc and or item movement upon board
* Game administration capabilities
* Game admin can kill a running game
* Game admin can add new players
* Game admin can update data of existing players
* Game admin can remove an existing player from a game.

# Difficulty scaling:

As each floor is generated the number of enemies increase increases with scaling health and damage. The quality of items will also increase as the player ‘dives’ deeper.

# Logging in:

When the user first loads the application, they will be taken to a log in screen where they are given the option to either log-in or register a new user. Should a user choose to sign in, their credentials are checked against the database. If the user chooses to register a new user, they create their username and password. When a player has a failed log-in attempt as in they input an incorrect username or password they are prompted with the amount of sign-in attempts left before they are locked out of signing in again for a period.

# Menu Screen:

After the user logs in with their account, they are automatically taken to the application's main menu screen, here two main options are displayed with those being ‘Play’ and ‘Options’.

# Options:

When the user selects the ‘Options’ menu from the main menu, they are shown the option to delete their account.

# Game lobby/Play:

Should the user press play from the main menu they are loaded into menu which has two buttons to choose whether to host a lobby or join an existing game. If the user selects ‘join existing game’ a popup for where the lobby code can be put will appear. If the user decides to create their own lobby they go into the game’s lobby room and their unique lobby code is displayed, as well as a display with the current users within that lobby.

# Gameplay:

A user(s) spawns onto the map on a dedicated ‘home’ tile starting from floor one, enemies will also spawn upon map generation and a way to advance to the next level via a door will also spawn. The goal is for the player to traverse the map, defeat monsters in the way of the exit and to pick up and find items to help them to advance to deeper levels, as the player advances to a deeper floor and kills more monsters they will gain experience which eventually will level up the player when a certain amount is reached, this level as well as the floor reached will determine the players score. The goal of the game is to reach the deepest floor at the highest level thus resulting in a higher score. While traversing the map the user can access their inventory and swap out their found items or use healing/buff items.

If multiplayer is to be selected the game handles/runs the exact same as if it were single player however the difficulty scaling will increase, and the players score’s will be joint and will be displayed as a co-op joint score.

# Administration:

Certain accounts will be designated as administrators meaning they have access to admin controls. These admin controls are unique to administrators and these controls are the ability to end a user's current run, add a new player to a current run, the admin may update or delete user's account information and the admin can remove a player from an existing game.

# Storyboarding:

The story boarding sketches for the game as shown below:

Storyboard one:

Choose log-in or register option screen



1.1: User selects the log-in option as is taken to storyboard two.

1.2: User selects the register option and is taken to storyboard four.

Storyboard two:

Login In screen



2.1: Player puts their username into the textbox.

2.2: Player puts their password into the password box, characters displayed as ‘\*\*\*’.

2.3: Log in button. Once pressed validates the username & password against the database to check for a match, if the user fails to input correct information pertaining to an account they will be given a username/password is incorrect prompt and the amount of tries left out of five will be displayed.

Storyboard three:

Username/Password incorrect



3.1: User presses ‘OK’ button to confirm and understand the failure upon attempting to log in. They are returned to storyboard 1

Storyboard four:

Account registration



4.1: Registration text to show the user they are creating a new account and not logging in.

4.2: The user Inserts the username they wish to have for their account.

4.3: The user inserts the password they wish to have for their account.

4.4: Create button to create the account with the given username and password. Once a user has pressed this button, they are moved to storyboard two.

Storyboard five:

Main Menu

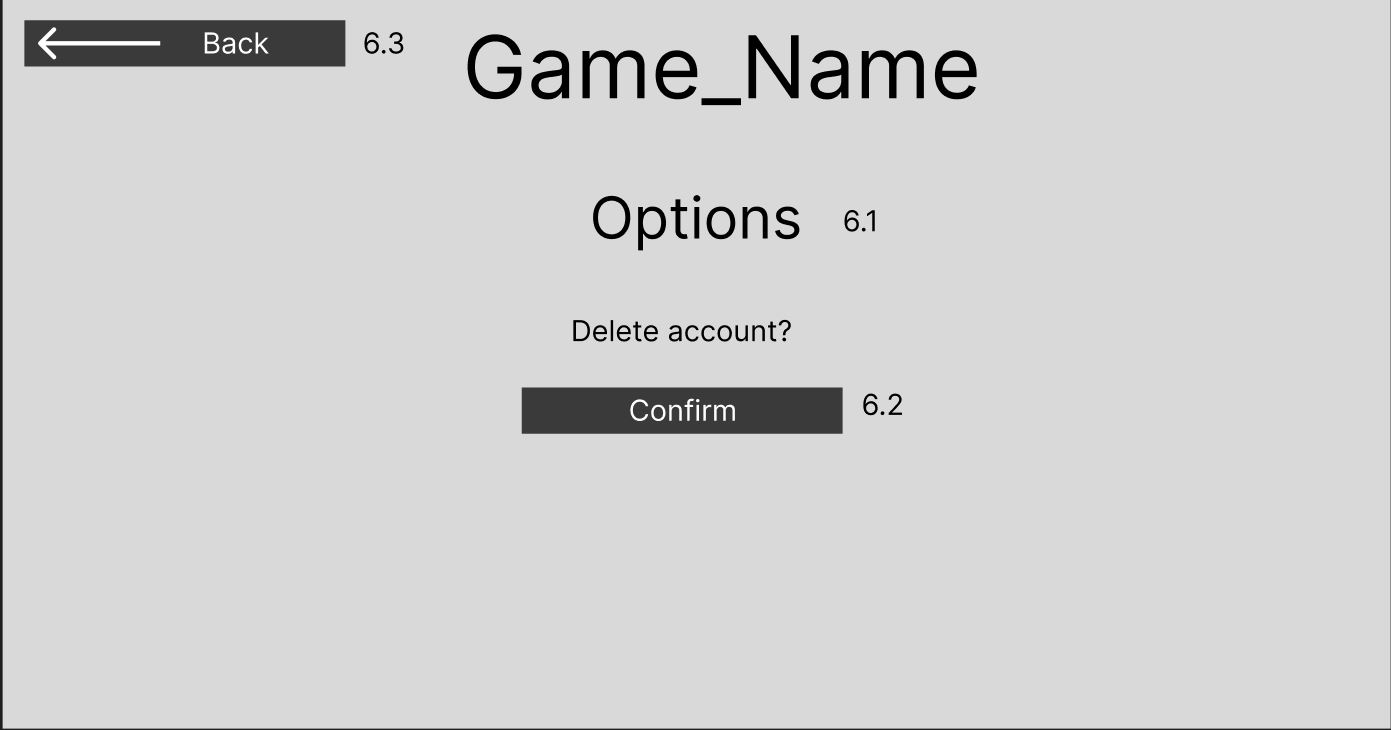


5.1: When the user selects the ‘play’ button the user is taken to storyboard seven and starts the game.

5.2: The user can choose to select the options menu from the main menu, when the options menu is pressed, they are taken to storyboard six.

Storyboard six:

Options menu



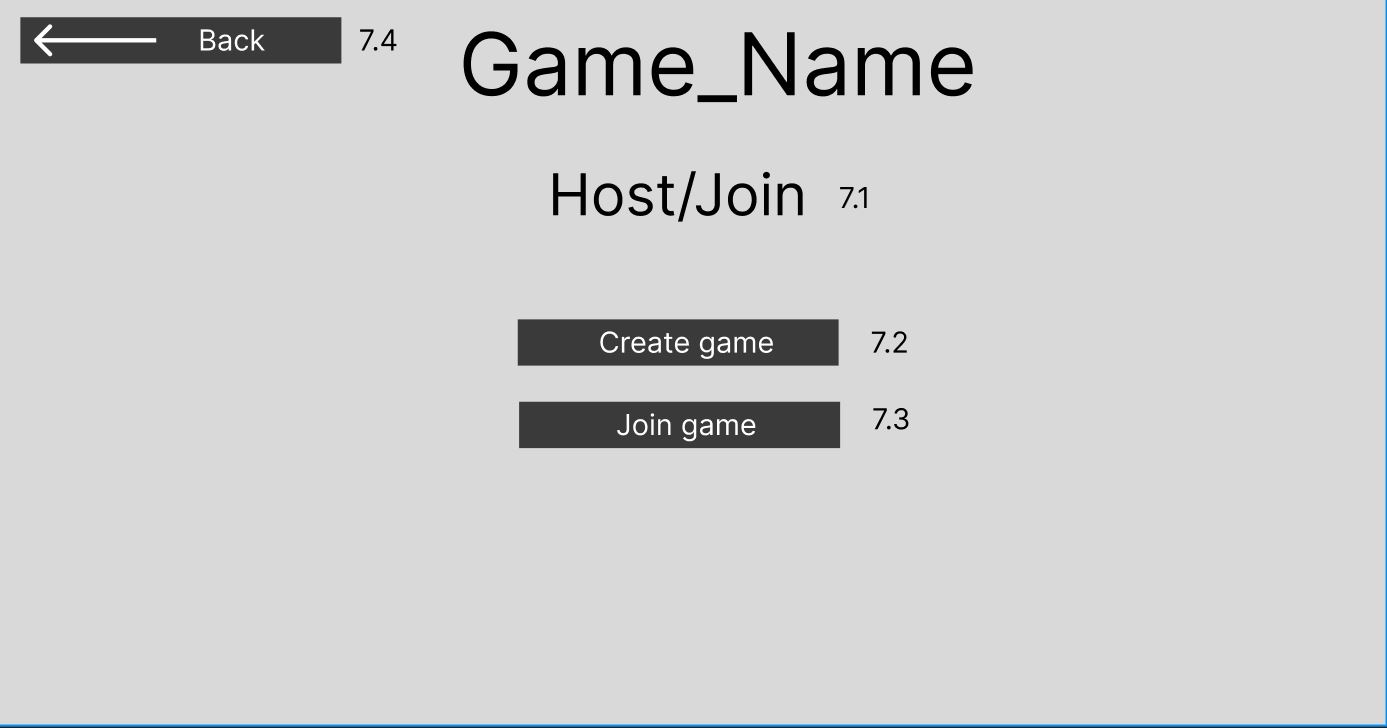
6.1: Options text to show user what page they’re on.

6.2: Confirmation to delete their account.

6.3: Back button to return to storyboard five.

Storyboard seven:

Lobby menu



7.1: Host/Join text to show user what screen they’re on.

7.2: Option for user to create a game.

7.3: Option for user to join an already created game via a code. Takes them to storyboard eight.

7.4: A back button to take a user back to storyboard five.

Storyboard eight:

Join game



8.1: Text for user clarification.

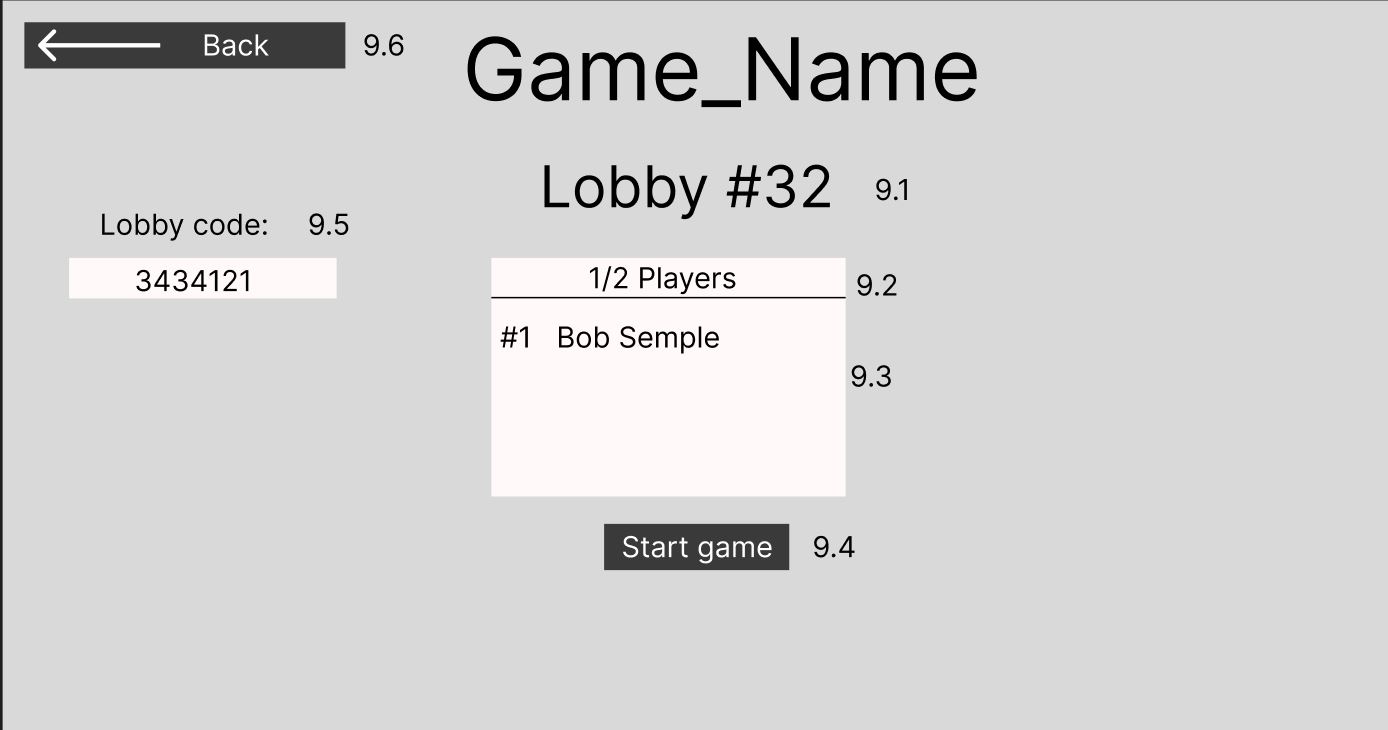
8.2: Text field for user to insert the lobby code.

8.3: Confirmation of code insertion.

8.4: A back button to take a user back to storyboard five.

Storyboard nine:

Create game



9.1: Displays the lobbies name.

9.2: Show’s the current users by quantity.

9.3: Displays current users within the lobby by name.

9.4: Allows the user to start the game.

9.5: Displays the lobby code for the user to give to another player.

9.6: A back button to take a user back to storyboard five.

Storyboard ten:

Administrator screen



\*Assumed you have logged in on an administrator account\*

10.1: Allows admin to play as normal

10.2: Allows admin to access options as normal.

10.3: Allows the administrator to access their admin control panel.

Storyboard eleven:

Administrator controls



11.1: Text to display the admin panel.

11.2: Allows the admin to insert the lobby code to edit a current game.

11.3: Allows the admin to confirm the code.

11.4: Allows the admin to input a user's account name to edit the account.

11.5: Allows the admin to confirm the code.

11.6: A back button to take a user back to storyboard ten.

Storyboard twelve:

Editing a game



12.1: Game edit text to show current page

12.2: Shows list of current players, click to kick player.

12.3: Button to end the game.

12.4: Text field to insert the lobby code into.

12.5: The admin can type a user’s username to add them to the game

12.6: A back button to take a user back to storyboard ten.

Storyboard thirteen:

Editing an account



13.1: Account edit text

13.2: An admin may edit a user's name.

13.3: Password cannot be seen/edited by admin.

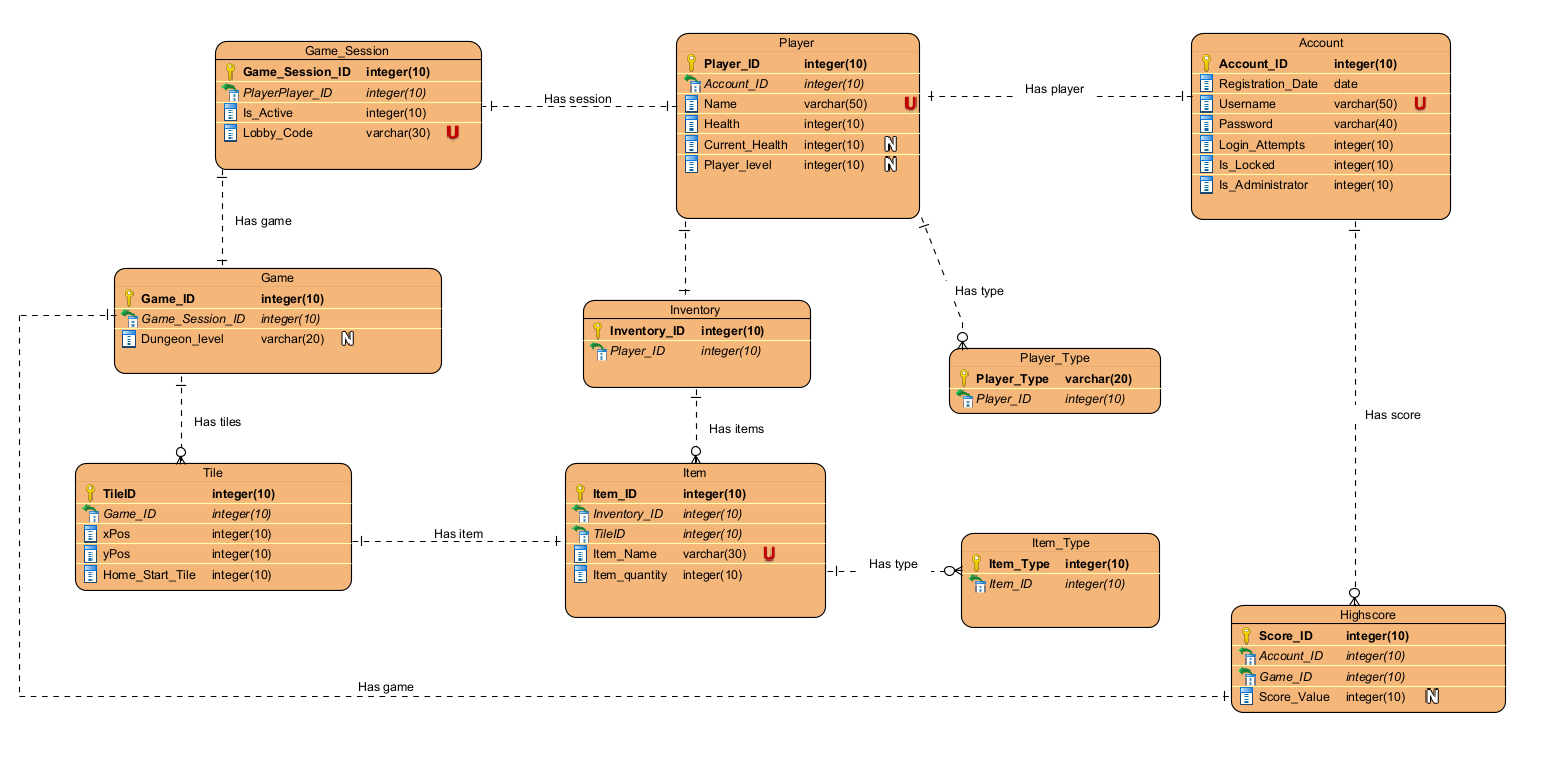
13.4: Allows the admin to delete player’s account.

13.5: Allows the admin to update user information written in the above text fields.

13.6: A back button to take a user back to storyboard ten.

## Entity relationship diagram

The ERD schematic file will be attached in the repository.



# Entity relationship diagram rationale

Account table:

Starting from right to left of the diagram, the account table contains information about the game user’s account such as their log in credentials, the number of times they’ve attempted to log in and if they’re an administrator. The account name is a unique value as different users must have different account names. An account has a score associated with it, one account can have many scores, through many different games.

High score table:

The high score table contains the information needed to link the score value to the corresponding game and account. Contains foreign key references to account and game tables. The high score table itself contains a score value. A game must have a high score and a high score must have a game.

Game table:

The game table contains the dungeon level, the game ID and a foreign key reference to the game session table. One game may have many tiles, and a game must have a game session.

Tile table:

The tile table contains all the information about the tiles on the game’s ‘board’. The tile table contains information such as the positions of the tiles as well as if the tile is a home start tile or not. A game can have many tiles, many tiles have one game.

Game session table:

The game session table contains information about the active game such as if the game is active or not (currently in use by players), the code used to enter the game, and the player ID’s of the player’s playing the game. The lobby code is unique as each individual lobby has a separate unique code to link it to a game.

Player table:

The player table contains all the information about the user’s in game character. The table contains the player’s statistics such as health, Player name, their level and their current health which may be different from their max health I.e. 30/50 health points. The player table has the most relationships of all the entities. The player’s current health and player level are nullable, as a player’s max health is just their health value and a player may not have a level.

Player type table:

There are multiple different player types, a player may be an NPC enemy or a user-controlled character. A player has many player types. The player type table contains the player’s ID and the type of player they are.

Inventory table:

A player has an inventory, the inventory table contains the player ID to associate the items within the inventory to them. An inventory has many items. Many items have one inventory.

Item table:

The item table contains the information about the items held within an inventory. The item table contains the foreign key references to the tile and inventory tables. An item must be on a tile, ‘tile has item’. The item table holds attributes for the item such as the item quantity and name of the item which is unique to that specific item.

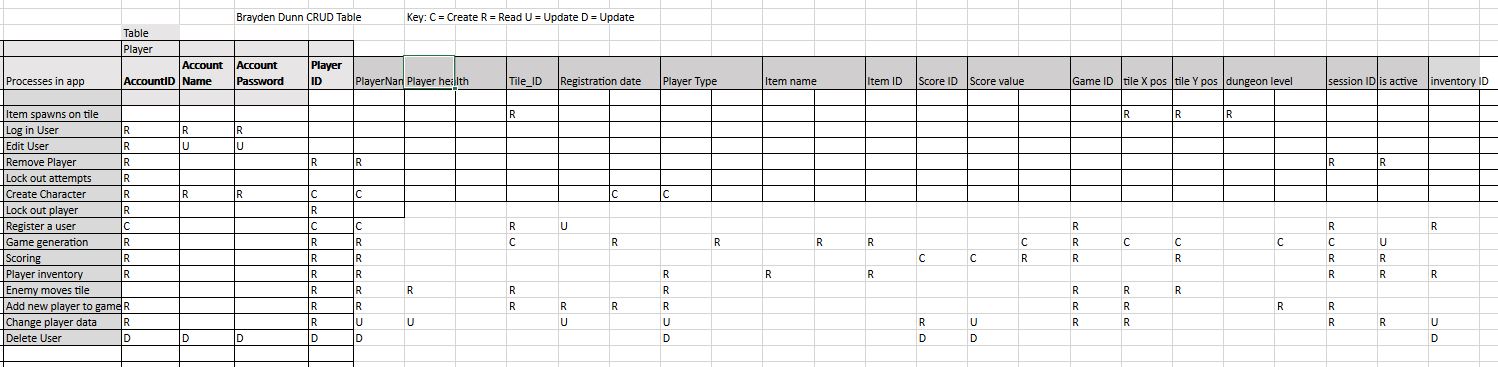
Item type table:

An item can have many types. An item type must be attributed to an item. The item type table defines the type of item it is I.e., a healing item or if it’s an equipable item such as a sword. The item type table contains the item type primary key and a foreign key reference to items ID.

# Crud Table:

(1. <https://www.educative.io/blog/crud-operations>) Within databases there exist operations which can be used to modify data; CRUD, short for create, read, update and delete commands are used to manipulate data and do as their name suggests it either creates, reads , updates or deletes the data. To interact with the database, you must use CRUD.

Within database planning a CRUD matrix can be used to understand how CRUD will be used within your application. Through the CRUD matrix you can visualize how the data being modified/accessed will be created, updated, read, or deleted when specific actions occur in the application. A CRUD matrix allows you to visualize the modification of the data when your application completes a function.



The excel spreadsheet will be submitted to the repository for easier viewing.

## DDL Script:

A DDL script is used to create the structure of the database, such as the tables and their relationships and restraints. It is used to describe the fields and records in a database.

My DDL script is called ‘dat602\_A1\_DDL’ , the DDL script will contain the creation of the database as well as the inserting of test data, based off my ERD.

**Add all files to a github repository with a brief readme overview of the repository!!!**

# References:

1. <https://www.educative.io/blog/crud-operations>

|  |  |
| --- | --- |
| Article title | CRUD operations explained: Create, read, update, and delete |
|  |  |

|  |  |
| --- | --- |
| URL | <https://www.educative.io/blog/crud-operations> |
|  |  |

|  |  |
| --- | --- |
| Website title | Educative |
|  |  |

|  |  |
| --- | --- |
| Date accessed | March 29, 2023 |

Written by: Maryam Sulemani