SEPTEMBER 4, 2024

FULL STACK GAME DEVELOPMENT

DAT602 ASSESSMENT MILESTONE 1

KIRA BYRNE
#13509995
NMIT

Table of Contents

INTRODUCTION

The Plot

DUST 2 DUST is a turn-based, grid-based, multiplayer fighting game set in the later age of the American frontier. The year is 1898 in the infamous town of Tombstone, Arizona. Each player is a gunslinger, fighting their way through law and lawlessness in the establishing wild west with only a Colt 45 revolver at their side. The player must defend themself through one-on-one shootouts with other players, collecting bullets around the map between fights where they can, ensuring they are equipped to handle the next fight before they are challenged!

Gameplay

Account & Menu

The player will make an account with an email, username, and password. They can log in with either the email address or username but the password will validate either.

The player can now log into their account where they can enter the game, read a tutorial that tells them the plot of the game and how to play, seek admin support which will open a mailto: link to the admin email (?), or quit the game.

The Player Character

Each player character is made equal upon starting a game. They have a revolver, 3 bullets, and 10 health points.

- The revolver isn't necessarily an item as listed below but a simple representation of a tool to use the bullets. It is only a name, nothing 'physical' within the game.
- A player is able to move around the map, tile by tile, and interact with the world upon clicking when in proximity to a tile of interest.
- As a player character survives the environment, they will score points. 30 points are given for every 30 seconds a player survives in the game.

Map

Upon starting a new game, the player will spawn into a 9x9 map on a home tile. They can move either by clicking the next tile (NSEW) or key stroking using the UDLR keys. The player will move around the map rather than the map moving around the player.

The map in itself will be a vector of tiles. 9x9 for each tiled map that is visible to the player, and 3x3 for the wider map the player can navigate upon reaching the edge of the map they currently reside on. Each tile will act as a coordinate identified by the primary key of each major tile of the map (i.e. the 3x3 of tiles).

NPCs & Items

Within each map will randomly spawn NPCs and items. Other players can login and wander around the map to find these items and NPCs.

Items the player can find:

- Bullets, needed for doing damage in duels.
- Whiskey, for replenishing the player's health score.

Scoring points

Players can earn points through different methods:

- The initial method is by staying in the game as long as they can without dying. For every 15 seconds a player is exploring the map, 10 points are added to the score.
- Players can earn extra points through finding items like bullets and whiskey.
- Time-based points stop accumulating during duels, however, winning a duel will allot the player 1000 points.

Dueling

Players can challenge each other to duels. When a player reaches up to 2 tiles away from another player will be able to left-click an enemy player to target them and an attack button to fire a bullet.

When a player fires a bullet (attacks), a cooldown debuff will apply which is a short window of time where the attacking player cannot attack again for 1.5 seconds.

- A timestamp of how long they survived in the game
- The final score earned before they died
- Their last high score if exists (select score where TOP timestamp ascending)

CRUD Analysis

DUST 2 DUST CRUD analysis.xlsx

Storyboards

1.0 Main menu

The first screen seen upon running the application client.

Functions

1.1 Should display the game title and three buttons.
1.2 Login into account
1.3 Create account
1.4 Exit application (closes window)

2.0 Account Sign-up

The account creation screen that allows players to enter criteria and create an account for gameplay.
Functions

Accessed through the 'sign up' button on the main menu.

2.1 – 2.5 Criteria requires:

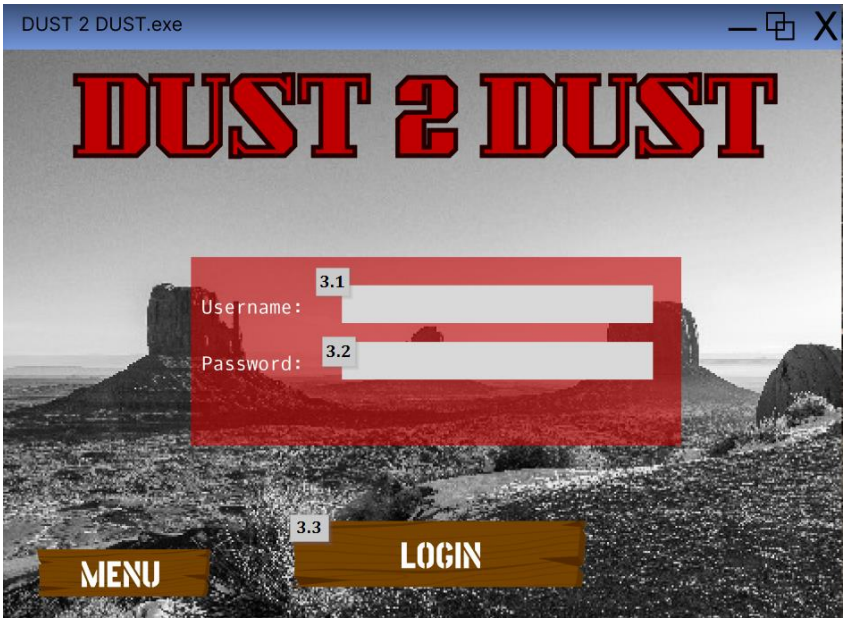
- Unique email
- Unique username
- Password

2.6 If the username the player is attempting to claim is already stored in the database, it will return this error.

2.7 If the player attempts to use an email address that is already stored in the database, it will return this error.

2.8 If the entered details meet the criteria requirements, a 'success' message will appear.

2.9 On clicking the 'sign up' button, all account details are submitted and stored in the database account page.



3.0 Login

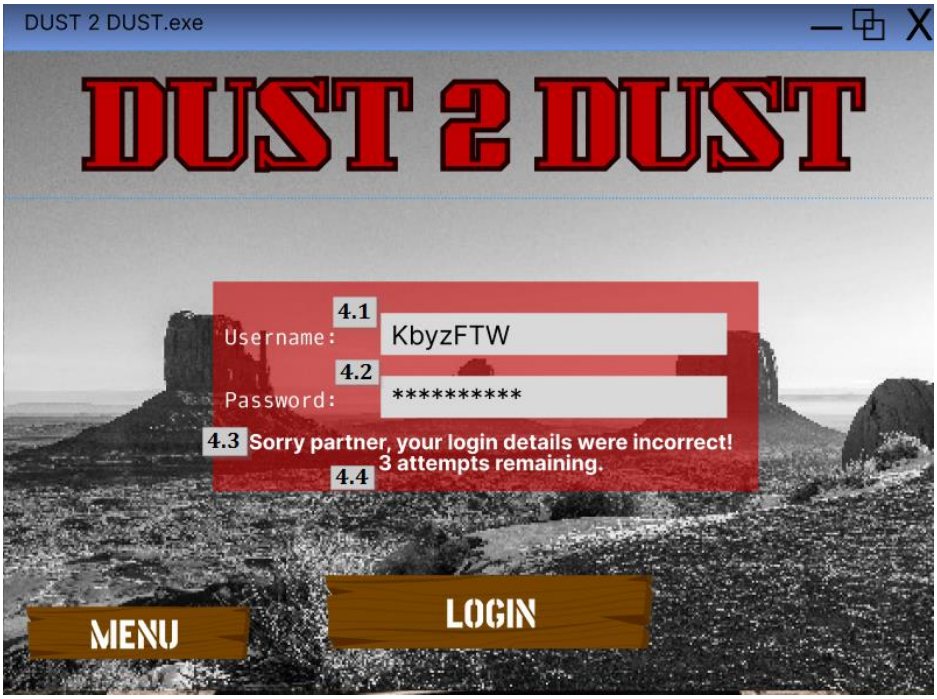
Allows players with a account stored in the database will be able to log into with their username and password.

Functions

3.1 Player will enter their username in the provided text box. This criteria will be checked against the list of usernames stored in the user accounts table in the database

3.2 Player will enter their corresponding password in the provided text box. This criteria will be checked against the password corresponding to the provided username stored in the user accounts table in the database. The password cannot work without a valid username.

3.3 When details are entered, the player can click the login button to attempt to login to their account. A login attempt will check the provided details against what is stored in the database.



4.0 Failed Login

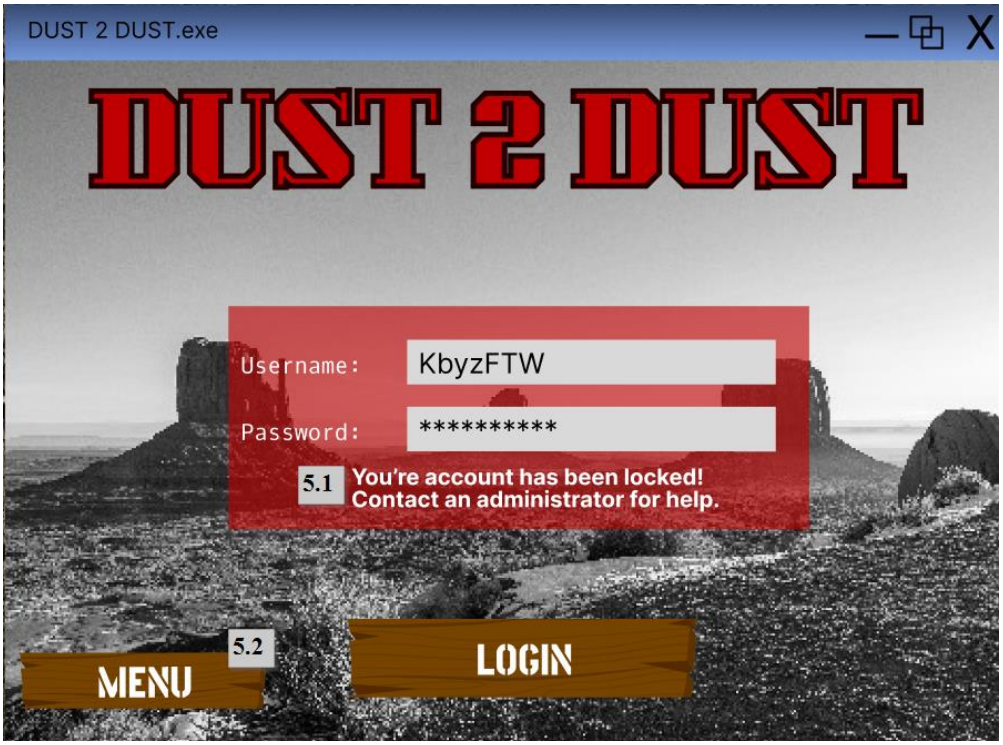
If a player attempts to login with invalid details, the login menu will produce an error message.
Functions

4.1 If the database has not stored the username provided in the textbox, it will return a login error regardless of the password (correct or incorrect).

4.2 If the database has not stored the corresponding password in the database, it will return a login error.

4.3 When a login attempt fails with the criteria in either 4.1 or 4.2, an error message will appear to notify the player of their failed attempt.

4.4 The database will store a number of login attempts, the maximum being 4 tries. Upon the next failed login attempt, the database will update the error message to display the number of remaining attempts.



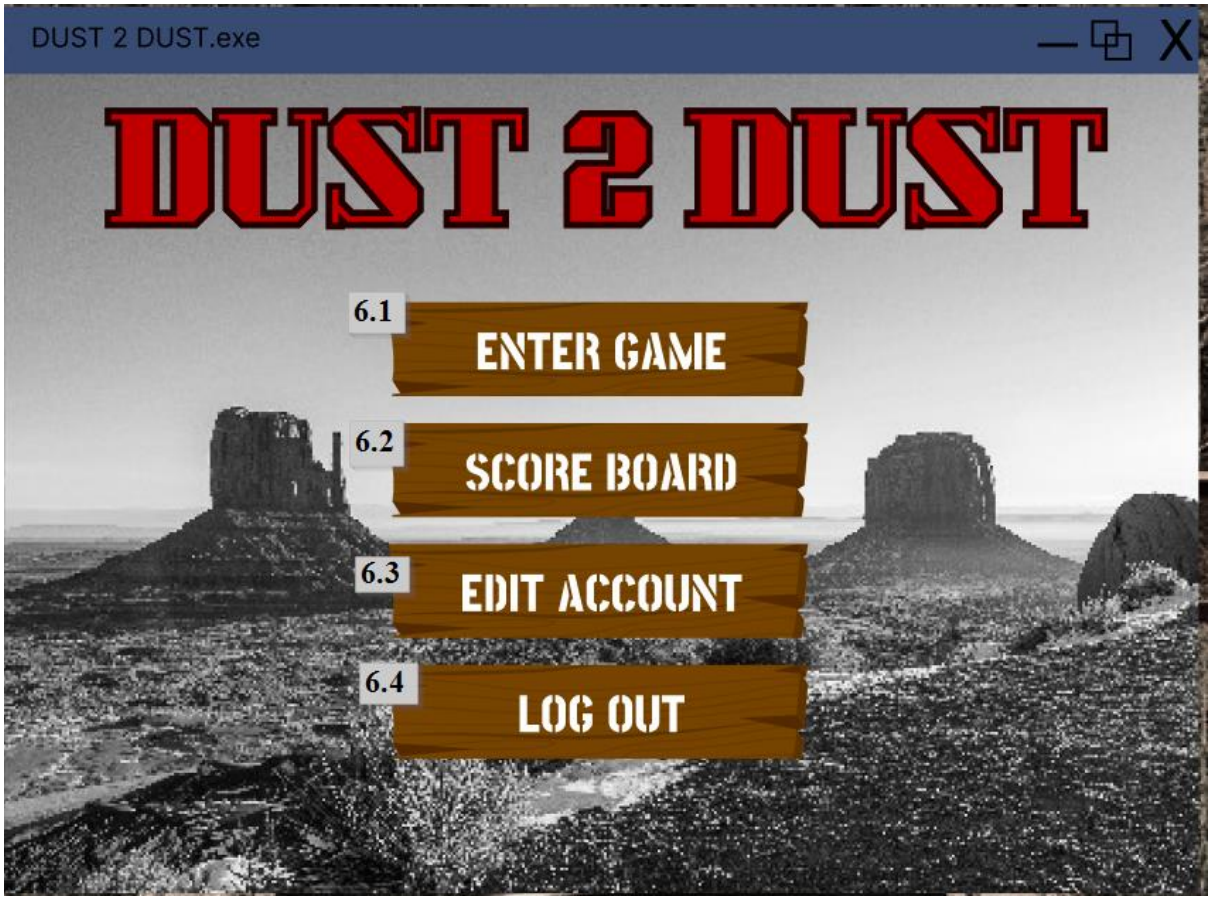
5.0 Account Locked

When a player has exceeded 4 login attempts with invalid details, their account will be locked until an administrator is contacted via the 'help' feature on the main menu.

Functions

5.1 After the fourth and final login attempt with invalid criteria, the error message will update to notify the player of their account being locked. The player will be encouraged to contact an administrator for help.

5.2 The player will be guided back to the menu via the main menu button where the 'help' option can be found.



6.0 Account Menu (logged in)

When a player successfully logs into their account, they will be taken to the account menu.

Functions

6.1 A player can enter a game. This will either create a game if there is no active players or return an active gameID to allocate the player to. All characters are created equally upon entering the game and will be identified through the account username.

6.2 View their scoreboard. This will show an aggregate of each player's highest score.

6.3 A player can edit their account details

6.4 Logout of their account. This will return the player back to the main menu as shown in fig 1.0.

7.0 Edit Account

A player can edit details in their account which will update any of the changed criteria. Functions. Accessed through the ‘edit account’’ button on the logged-in menu as seen in fig 6.

Criteria requires:

7.1 Unique username

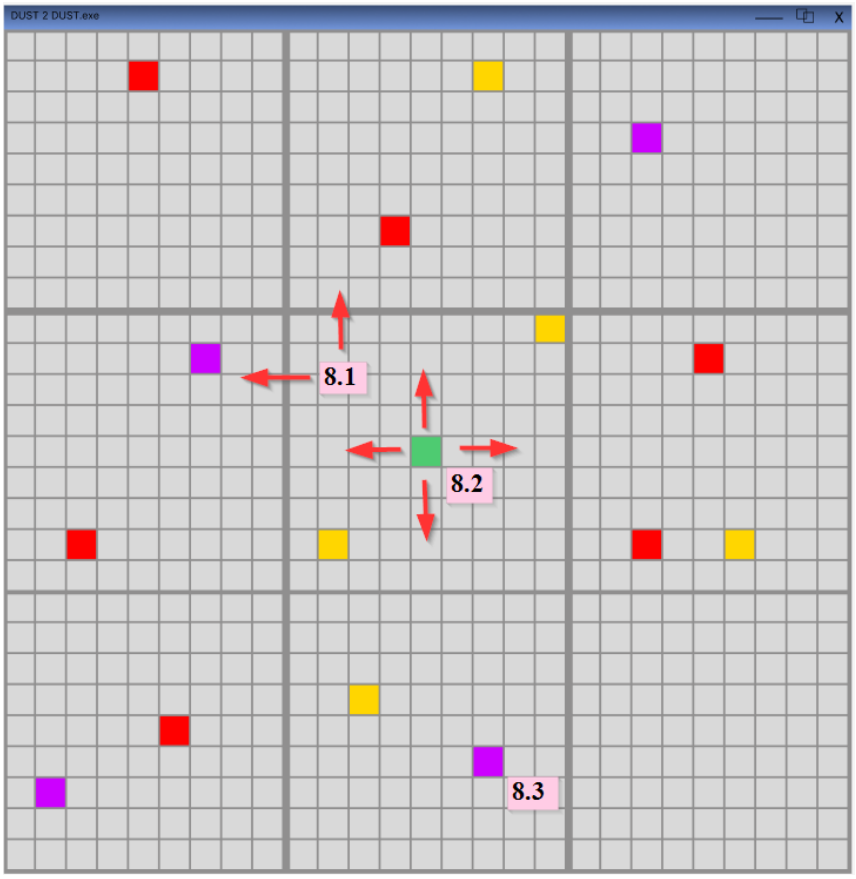
7.2 Old password to confirm the account holder is valid

7.4 New password

7.5 Confirmation of new password

Players cannot change their registered email autonomously. This will be handled through a request email to the administrator.

On clicking the ‘save’ button, all edited account details are checked for the above criteria in the database. If criteria are met, the new details are saved.



8.0 Whole Game Map - 3x3 Vector Grid

This is a visual of the entire map that exists in a game. The entire map (NOT seen by the player) is a 3x3 grid of 9x9 grids. When a player is active, they will only view the 9x9 grid they are actively on.

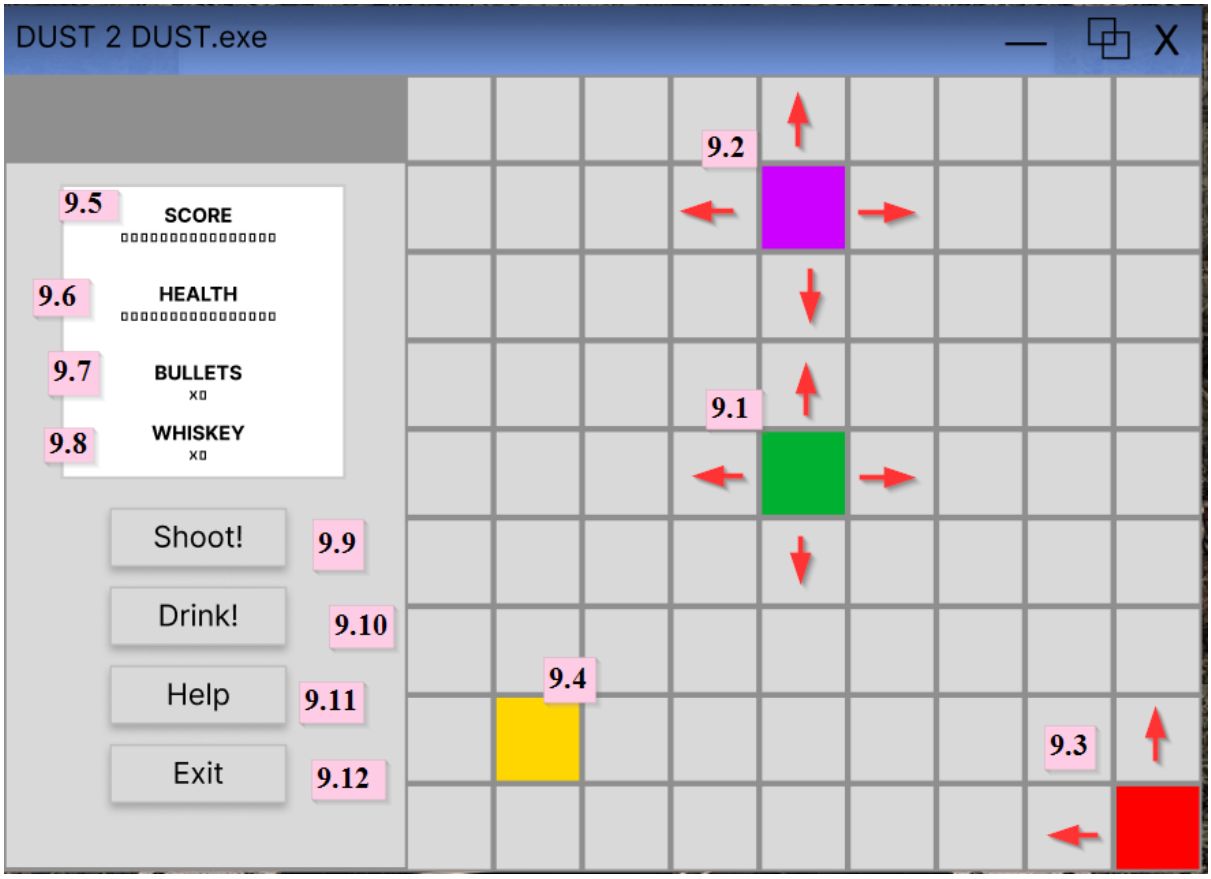
A game will have active assets across all grids. It will be programmed to randomly spawn and de-spawn items and NPCs after players have interacted with them of after allotted amounts of time.

Functions

8.1 A game exists within the bounds of the 3x3 map.

8.2 A player will be able to view and navigate one 9x9 grid within the map. The tile coordinates will be identified by each gridID within the map i.e.
gridID: ‘west2’ tileID: “-1.0, 0.2”

8.3 A new game will generate items on commencement All items and NPC will load across the map at random intervals. All players in a game can see all items, NPCs and other players.



9.0 Game Screen – 9x9 grid

As mentioned in fig 8.0, the map is a 3x3 vector of 9x9 grids. Seen in above (fig 9.0) the 9x9 gird represents the part of the 3x3 map visible to the player which they will be able to navigate to find the NPCs, items, and enemy players as shown in the figure.

Functions

9.1 A player will spawn on a random empty tile (coordinate) in a random grid (gridID) in the map. They will be able to navigate on the X and Y axis using the keyboard keys up, down, left, and right as represented by the red arrows. Upon reaching a boarder, they will move to a near by grid if available. If not, they will remain still.

9.2 When a game is created, Non-player Characters (NPCs) will spawn throughout the map which are visible to all players. NPCs are able to move via the database updating their location every few moments.

9.3 Enemy players are represented as red tiles and have all the same functions as the player does.

9.4 Items, represented by yellow tiles, will randomly appear on the map to be interacted with by any player.

9.5 – 9.8 A side bar on the game screen will display the players current stats and items including:

> 9.5 Their current score. The score will allot 100 points for every 30 seconds the character spends in the game. The primary goal is to survive as long as possible.
>
> 9.6 Their current health. Each character is given full health displayed as 10/10 ‘points’. When a player’s health reaches 0, they will die.

9.7 The amount of bullets they have collected. All characters will start with 6 bullets but this number is limitless.

9.8 The amount of whiskey they have collected. Whiskey can be used to restore health points. Players can find whiskey throughout the game from item tiles or NPCs. Players will begin a game with no whiskey stored.

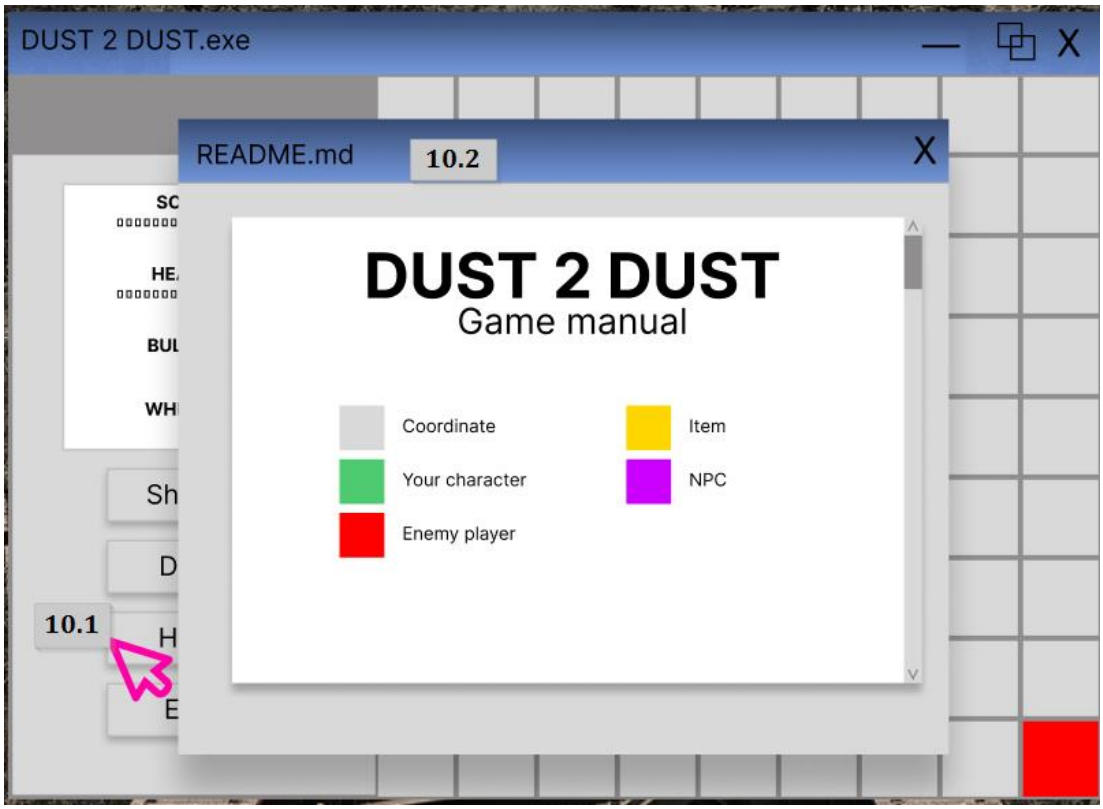
9.9 – 9.12 The player can interact with game mechanics through a collection of buttons on the side bar including:

9.9 The 'shoot!' button will allow them to attack an enemy player that is targeted through a left-click event. Clicking this button will retrieve and update the quantity of bullets the player has. The button can only function when bullet quantity is >=1.

9.10 The 'drink!' button allows a player to use a bottle of whiskey to restore health. Clicking this button will return and update the quantity of whiskey the player has. The button can only function when whiskey quantity is >=1 and health points are <10.

9.11 The player can access the game manual from the 'help' button.

9.12 The player can exit the game. This will remove their character from the active game and return them to the account menu.



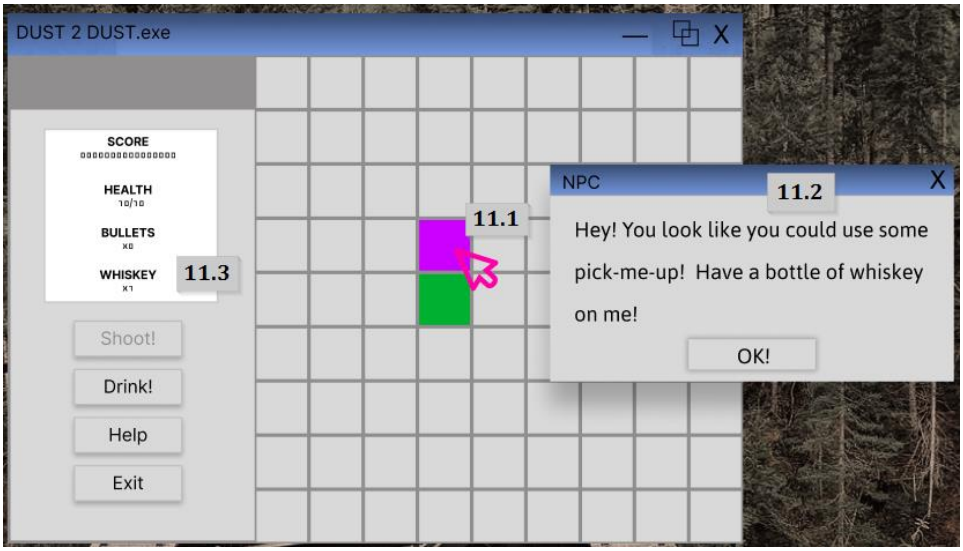
10. Game Help Menu

The player can access the player guide/game manual via the help button to read information about the game. This will not pause the game but open an independent window.

Functions

10.1 The help window is accessed via a left click on the ‘help’ button in the left hand side bar.

10.2 The help window will open a rich text file where the game guide can be viewed.



11. NPC Interaction

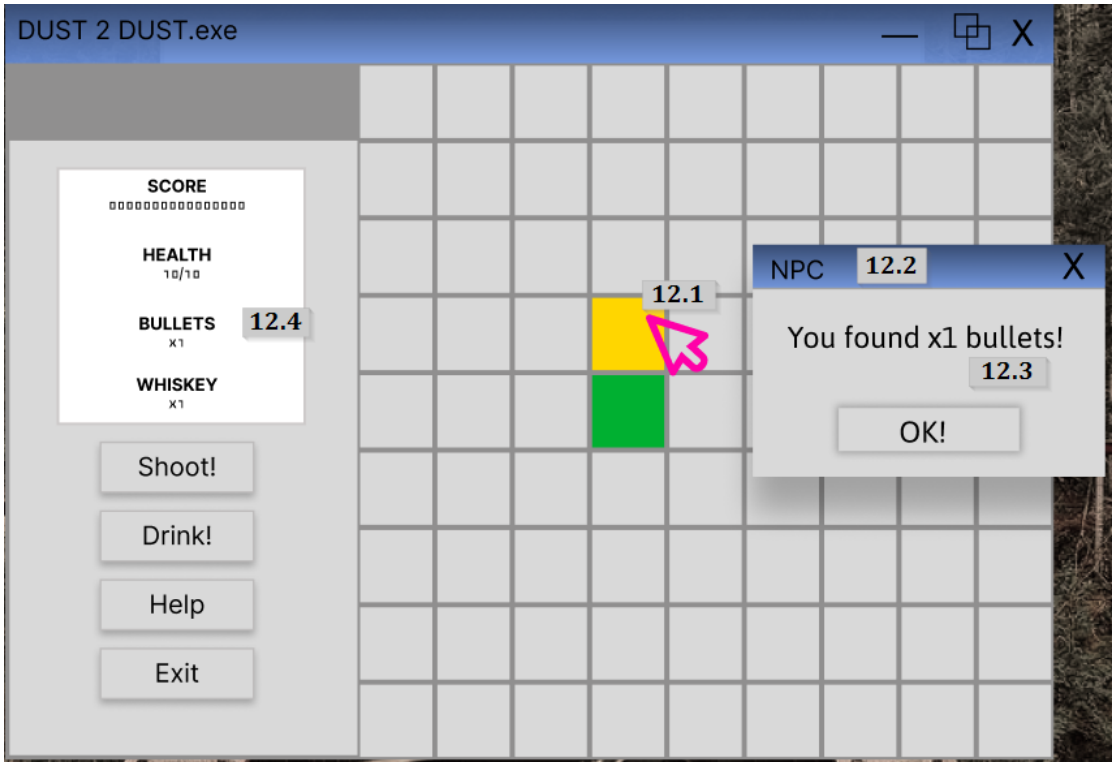
Upon approaching an NPC by standing on a bordering tile, a player can left-click the NPC to interact with them. This will open a dialogue window where the NPC may “say” something to the player or give them an item on random chance. After interacting with a player, the NPC will ‘de-spawn’, leaving their tile empty.

Functions

11.1 Once a player is standing on a tile that borders an NPC tile (purple), they can left-click the tile to interact with it.

11.2 Interacting with an NPC will open a window with dialogue text. The dialogue will be randomised and may have the chance to give the player an item.

11.3 Item given by NPCs will show in the player’s inventory.



12. Item Interaction

When a player is bordering an item tile (yellow) they can left-click to interact with it. This will open a window that will show text regarding what was found and how much. A random quantity and item name from the database. Once a player has interacted with an item tile, it will add the item(s) to their inventory and 'de-spawn' becoming an empty tile.

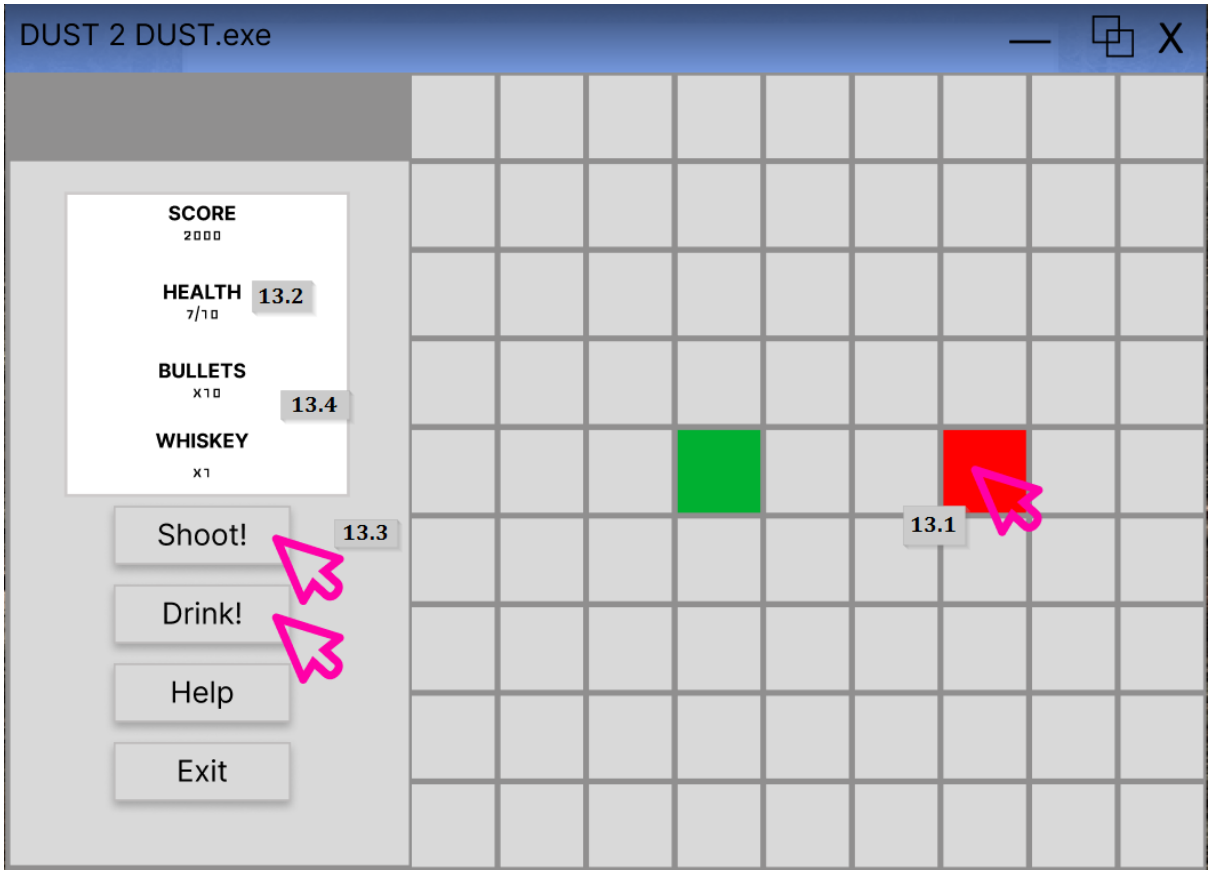
Functions

12.1 The player interacts with an item tile by bordering it then left-clicking it.

12.2 Upon interaction, a pop-up window appears with text about what the player has found.

12.3 The game will pull an item name and randomise a quantity to allot to the player.

12.4 The player's inventory will update the quantity of the item found joining the item data from the interaction window to the inventory table.



13. Dueling Enemy Players (Combat)

Other players will be considered enemies and can engage in combat. A player will be able to 'kill' another player when they at least two tiles away from the player's position.

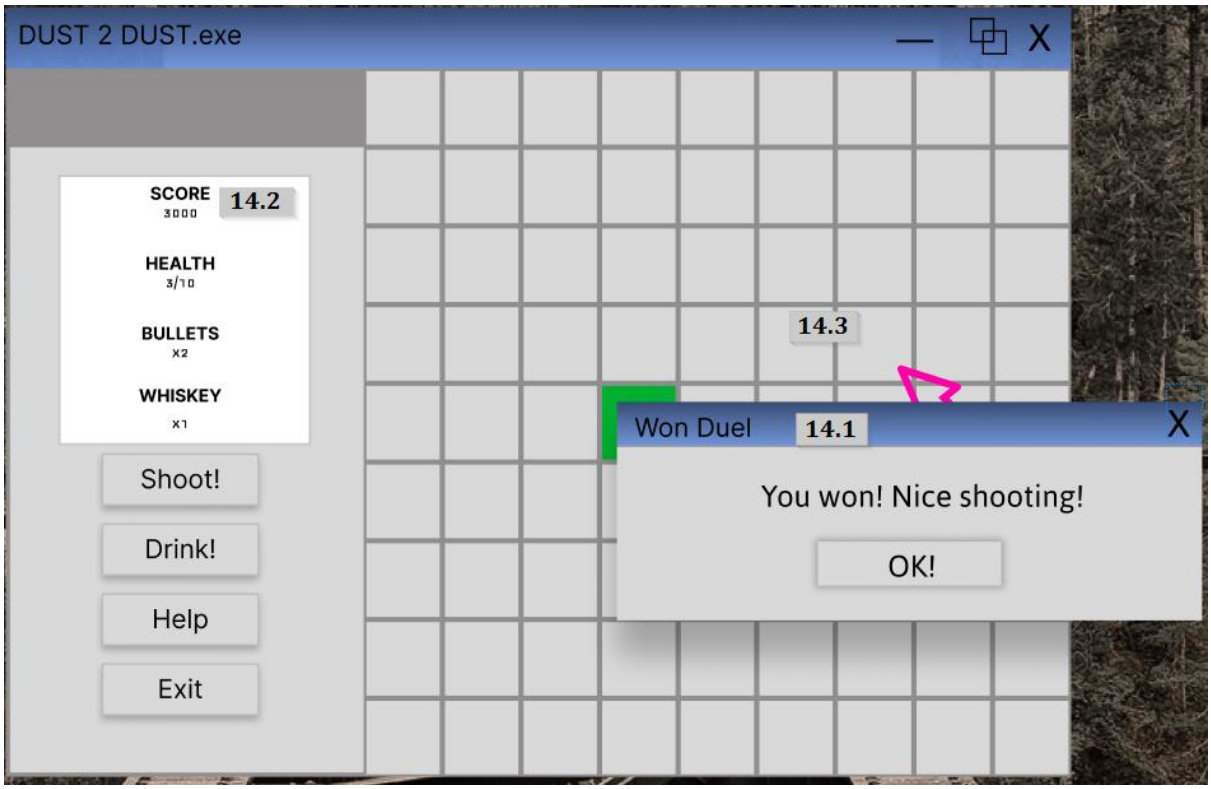
Functions

13.1 Select an enemy player (=< 2 tiles distance) with a left-click.

13.2 Taking damage will decrease the total health stored in the character table.

13.3 The player can deal damage with the 'shoot!' button or restore health with the 'drink!' button

13.4 Utilising the damage or health buttons will update the quantities of their respective items in the player inventory. Upon reaching 0, the buttons will become ineffective.



14. Combat Won

If the player is able to reduce the enemy player's health to 0/10, the enemy player will 'die', resulting in a won duel for the player.

Functions

14.1 Return window displaying 'win' message
14.2 Allot 1000 points to the winning player
14.3 Remove the enemy player from the game.

15. Combat Lost (Death)

A player loses a duel when their health is reduced to 0 through combat (enemy bullets). They will be shown a 'death' window and removed from the game.

Functions

15.1 Remove the player from the game, returning them to the account menu.

15.2 Display the 'death' window.

15.3 Return the player's score on death of the last game and their highest score previously earned.

16. Scoreboard (Public)

The public scoreboard is an ascending list of the top 10 players with the highest scores. Players can access the public scoreboard via the ‘scoreboard’ button on the account menu.

16.1 The scoreboard joins the username to their highest score.

16.2 The scoreboard list the top 10 highest scores ascending.

17. Administrator Account Menu

Administrator accounts have a administration level functions displayed in their logged-in menu.

17.1 The administrator has access to the administrator settings which will return live-game data in an interactive window.

18. Administrative Settings

An administrative settings window allows administrative accounts to see all active game activity including active games by ID and players in each game by username. Administrative accounts have the ability to kill active games, remove players from a game, or ban a player.

18.1 The administrative window opens via the ‘admin settings’ button in the administration account login menu as seen in fig 17.

18.2 All active gameIDs are listed in the active games bar. The admin can select a game via a left-click to access more details or manage that game.

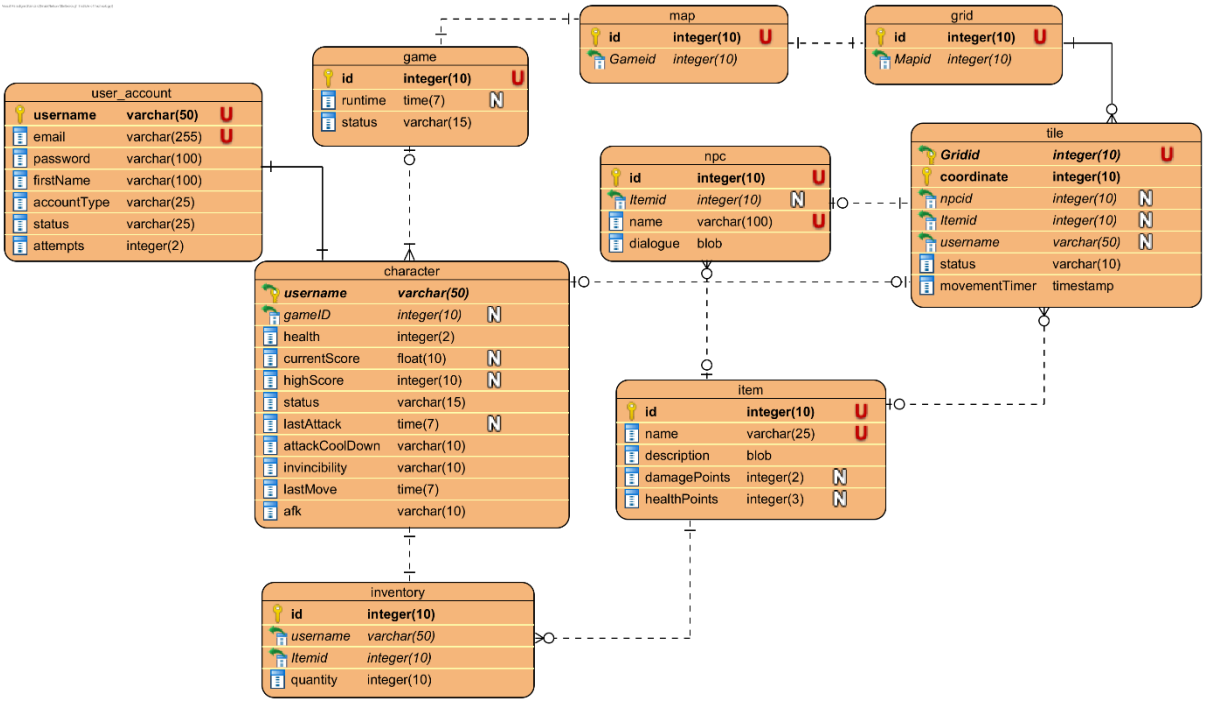
18.3 An admin can ‘kill’ a game which will remove that active gameID from the database and remove each player, sending them back to the login menu.

18.4 An admin can see a list of all player usernames that are currently playing an active game selected from the active games list. They can left-click a player’s username to highlight them for further management.

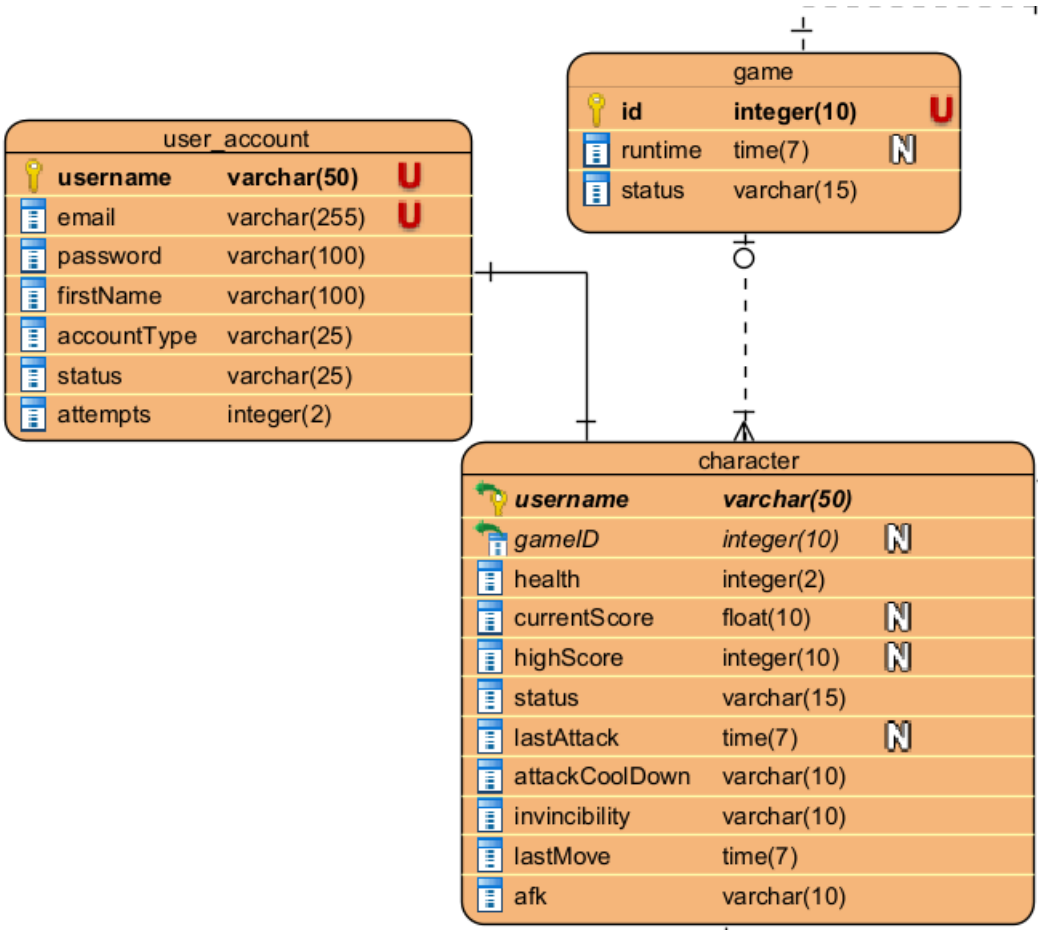
18.5 An admin can remove a highlight player from their game, returning them to the login menu.

18.6 An admin can ban a highlighted player from DUST2DUST, which will delete their account information from the database.

Entity Relationship Diagram

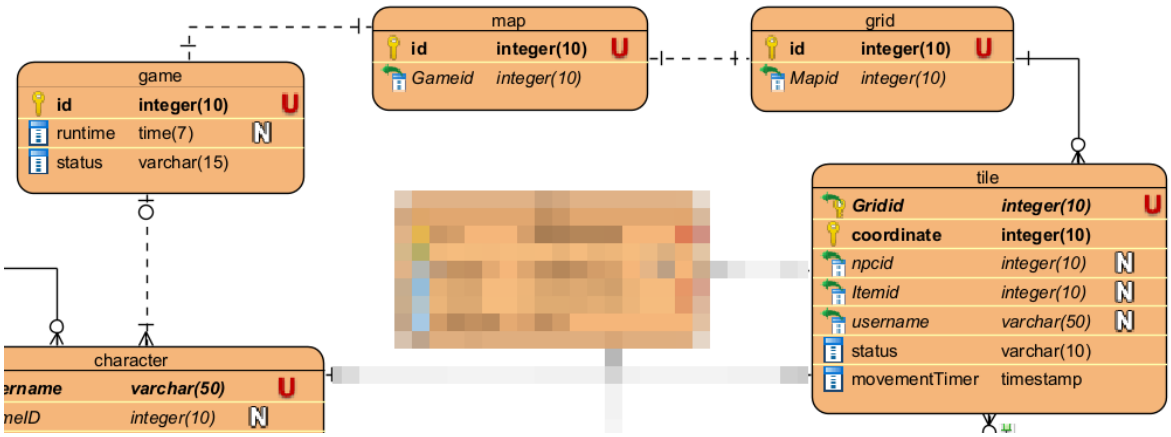


User accounts & Player Characters



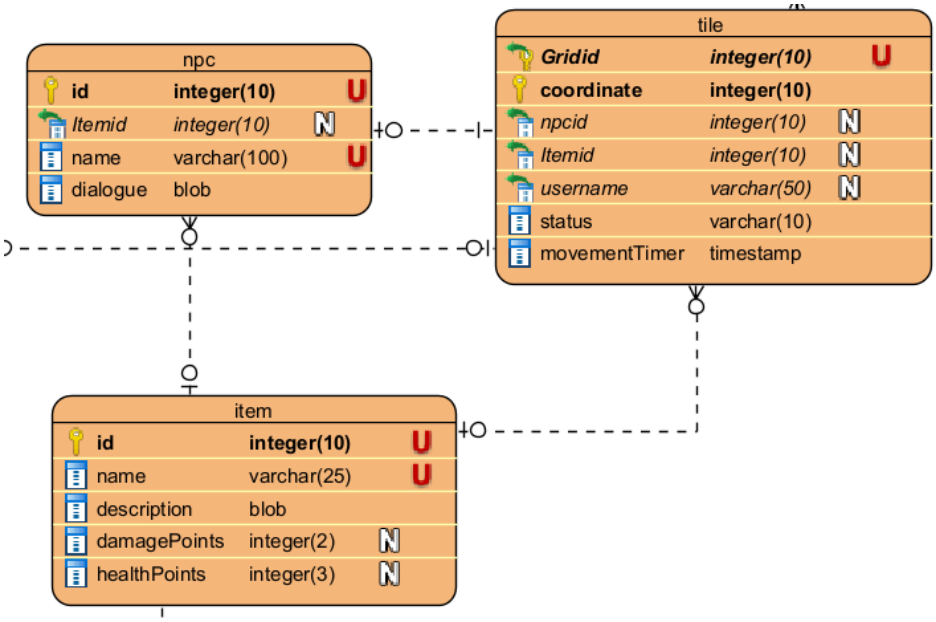
1. The user account tables stores account and account login information including:
    a. Username – The primary key identifier of a user account which users can create themselves with up to 50 characters. As a primary key, each entry of a username must be unique.
    b. Email - A unique email address supplied by the user. This will be used to contact a user from an administrator account.
    c. Password - The account's password. A string of up to 100 characters that will act with the username as a key to logging into an account.
    d. firstName – The first name or preferred name of the account user. A business rule of DUST2DUST is that only first name's are necessary for communication formalities between administrators and standard users.
    e. accountType – Set as either a standard player or administrator account. The text stored in this field will be checked against a constraint to ensure that the correct account type label is entered that will set the role and abilities the account has.
    f. Attempts – Stores the amount of login attempts the user has made. When the integer reaches === 3, the status of the account will be set to LOCKED.
2. A user account will be able to join the game on one character. This table will store active data on player characters, updating specific fields during game play.
    a. Username - This character will be known by the account username, acting as both a foreign key and primary key.
    b. gameID – This is a nullable foreign key displaying the id of a game the player character may currently be active in.
    c. health – Stores and updates an integer that represents the player's health points in the game. These will be lost and gained during action in a game.
    d. currentScore – Stores and updates the current accumulated score the player has earned by surviving in the game and winning duels.
    e. highScore – Stores the highest score a player has earned. This will be queried on the public scoreboard and the character death screen.
    f. status – A check constraint text which displays the player's status as Active or Offline (in a game or not in a game).
    g. lastAttack – When a player makes an attack move, they will be allotted a cooldown timer before they can attack again. This field stores the runtime of the game as HH:MM:SS when the player made their last attack move.
    h. attackCooldown – This is a check constraint field that checks the time of the lastAttack field and applies a momentary 'cooldown' as true or false. True will disable the player from being able to attack for an amount of time. After that time as passed from the lastAttack field, the cooldown will become false, enabling the player to attack. This is to prevent spamming of damage between players.
    i. Invincibility – A check constraint field that applies an invincibility status as true or false. When a player is attacked, this field will become true, making the player immune to damage for a short time. This is to prevent being damaged by multiple players at once too quickly.
    j. lastMove – A time check field that will update with the runtime stamp of the game every time a player makes any movement in the client.
    k. Afk – A check constraint field that works with the lastMove field to update the status as true or false. This field will default to false, but if a player does not interact with the game within a certain amount of time, the afk field will become true, removing the character from the game.

Game, Maps, Grids, & Tiles



1. The game table will store information about each game run in the server. A game cannot exist until at least one player enters. Upon first player entry, the game table will generate:
    a. GameID - A new gameID to identify that game instance as a primary key.
    b. Runtime - Begin a runtime of that game stored as a HH:MM:SS.
    c. Status - Set a default status of 'Active' which can be quired by the admin accounts to see all active games. This status will update to 'Inactive' when a game has no players or is 'killed' by an admin.
2. When a game is created, the map table will store information on the 3x3 vector grid of grids that will be generated to represent the whole map or boundaries of the game. This will store an ID for the entire map relating to a game via the gameID foreign key.
3. The mapID will foreign key into the grid table which will store information on the 9x9 grid blocks that make up the 3x3 wider map. Each 9x9 grid will be identified with a primary key.
4. The gridID will foreign key to the tile table which will store information on each tile in a grid block including:
    a. GridID/Coordinate - The gridID and coordindate which will act as composite keys to identify both the coordinate and the grid is it related to. Each tile will have a coordinate between 0.0 and 9.9 before reaching the boarder of the next grid.

NPCs, Items, & Tiles

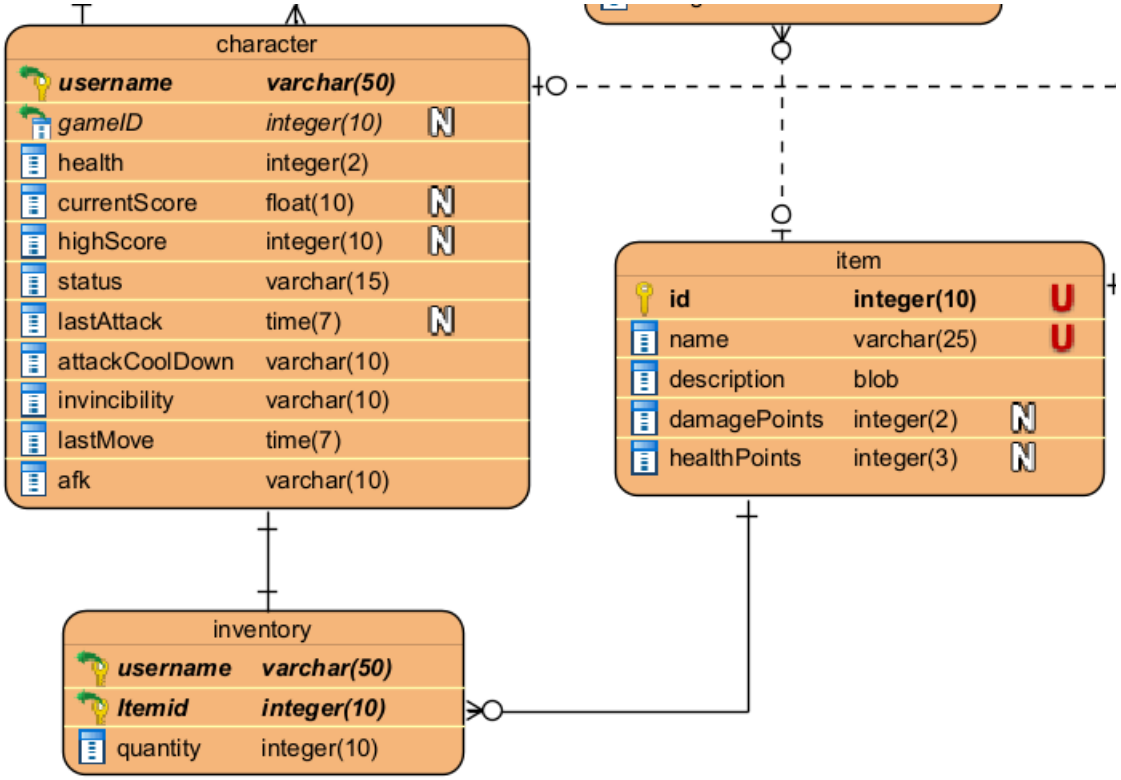


1. The tile table also stores data on the assets that could occupy a coordinate including:
    a. npcID- A nullable NPCID which will foreign key from the npc table to retrieve relevant data if a tile becomes an NPC.
    b. itemID - A nullable itemID which foreign keys from the item table to retrieve relevant data if a tile becomes an item.
    c. Username - A nullable username which retrieves data from the character table to identify both where a player is to the player themselves and others.
    d. Status - A check constraint field that will update the status of a tile as 'empty' or 'taken'. This status will determine a player's ability to move onto a tile.
    e. movementTimer – When a tile is checked as containing npc data, a timer stored at HH:MM:SS will be used to update the NPC's position every few seconds to show movement across the grid.
2. The NPC table stores pre-defined and generated information on Non-player character (NPC) assets that can be generated into a game and interacted with by players including:
    a. npcID – A unique identified for each NPC which will foreign key into the tile table.
    b. itemID – Some NPCs may give an item to the player when interacted with. The itemID is a nullable foreign key which will return the primary key of an item for the NPC to give but will be nullable as not all NPCs will give items.
    c. Name – The full name of the NPC stored as varchar text e.g. *Wyatt Earp.*
    d. Dialogue – A blob object that stores a string of long text that will be returned by the dialogue window that appears when a player interacts with a character.
3. The item table stores pre-defined data on items that can occupy tiles, be given out by NPCs, and collected/used by players. The data stored includes:
    a. itemID – A primary key to uniquely identify each item implemented in the game.
    b. Name – A meaningful name given to the item that will be displayed in the player's GUI (e.g. Bullet).
    c. Description - A short description stored as blob text which will be shown in the item window, illuding to its use in the game.
    d. damagePoints – A nullable integer which allots the amount of damage points the item can deal. This number will be subtracted from the enemy player's health field

when damage is dealt to them through player combat. Not every item can deal damage.

e. healthPoints – A nullable integer which allots the amount of health points the item can restore to the player character's health field on use. Not every item returns health.

Character Inventory & Items



1. The inventory table will store data on items that a player has collected and used throughout the gameplay.
2. A field within an inventory is identified by a composite key (Ravikiran, 2024) of the the foreign key itemID which will identify each item in the character's inventory and the username foreign key of the character who owns the inventory.
3. All items will exist in the player's inventory with a default integer of 3 bullets and 0 whiskeys. The quantity will update when items are collected or used during gameplay.

MySQL DDL

Drop Schema, Drop Tables

```
1  DROP SCHEMA IF EXISTS dust2dust;
2  CREATE SCHEMA dust2dust;
3
4  USE dust2dust;
5
6  DROP TABLE IF EXISTS `tile`;
7  DROP TABLE IF EXISTS `inventory`;
8  DROP TABLE IF EXISTS `item`;
9  DROP TABLE IF EXISTS `npc`;
10 DROP TABLE IF EXISTS `grid`;
11 DROP TABLE IF EXISTS `map`;
12 DROP TABLE IF EXISTS `game`;
13 DROP TABLE IF EXISTS `character`;
14 DROP TABLE IF EXISTS `user_account`;
15
```

1. A collection of statements that will drop the entire existing schema /database used to run the entire script from scratch.
2. Create the schema / database.
3. Use the created schema / database.
4. Drop each table created in the existing database to run the script from the beginning without the constraints of foreign keys and dropping tables in use (IBM, 2024).

Creating the Game & Map

```
23
24 CREATE TABLE `game`(
25     `gameID` INT PRIMARY KEY,
26     `runtime` TIME NULL,
27     `status` VARCHAR(10) NOT NULL
28 );
29
30
31 CREATE TABLE `map`(
32     `mapID` INT PRIMARY KEY,
33     `gameID` INT,
34     FOREIGN KEY (`gameID`) REFERENCES `game` (`gameID`)
35 );
36
37
38 CREATE TABLE `grid`(
39     `gridID` INT PRIMARY KEY,
40     `mapID` INT,
41     FOREIGN KEY (`mapID`) REFERENCES `map` (`mapID`)
42 );
```

1. Creating the game table, setting the gameID as the primary key.
2. Creating the map table, setting the mapID as the primary key and the associated gameID as a foreign key.
3. Creating the grid table, setting the gridID as the primary key and the corresponding mapID as a foreign key.

User Account

```
CREATE TABLE `user_account`(
    `username` VARCHAR(50) PRIMARY KEY UNIQUE,
    `email` VARCHAR(255) UNIQUE,
    `password` VARCHAR(100),
    `firstName` VARCHAR(100),
    `accountType` VARCHAR(25),
    `status` VARCHAR(25),
    `attempts` INT (3)
);
```

1. Creating the user account table with the username as the primary key and the associated criteria with unique constraint on the email address to prevent re-use of an existing email (W3 Schools, 2024).

Character (Player)

```
CREATE TABLE `character`(
    `username` VARCHAR(50) PRIMARY KEY,
    `gameID` INT NULL,
    `status` VARCHAR (10),
    `health` INT(4),
    `currentScore` INT(10),
    `highScore` INT(10),
    `lastAttack` TIME,
    `attackCooldown` VARCHAR (10),
    `invincibility` VARCHAR (10),
    `lastMove` TIME,
    `afk` VARCHAR (10),
    FOREIGN KEY (`username`) REFERENCES `user_account` (`username`),
    FOREIGN KEY (`gameID`) REFERENCES `game` (`gameID`)
);
```

1. Creating the character table which will use the username foreign key as the primary key to identify the player and the gameID foreign key of an active game they may be in. This nullable for when a character is not in a game.

Items & NPCs

```
91  CREATE TABLE `item`(
92      `itemID` INT PRIMARY KEY,
93      `itemName` VARCHAR(25),
94      `description` TEXT,
95      `damagePoints` INT(2) NULL,
96      `healthPoints` INT(2) NULL
97  );
98
99
100 CREATE TABLE `npc`(
101     `npcID` INT PRIMARY KEY,
102     `npcName` VARCHAR(100),
103     `dialogue` TEXT,
104     `itemID` INT NULL
105 );
```

1. Creating the item table which will identify each item by the itemID primary key.
2. Creating the npc table which will identify each npc by the npcID primary key. The itemID foreign key will provide an item to some NPCs who are design to give items to a player character.

Map Tiles

```
110 CREATE TABLE `tile`(
111     `coordinate` DECIMAL (19,0),
112     `gridID` INT,
113     `npcID` INT NULL,
114     `itemID` INT NULL,
115     `username` VARCHAR(50) NULL,
116     `status` VARCHAR (10) NOT NULL,
117     `movementTimer` TIME,
118     PRIMARY KEY (`coordinate`, `gridID`),
119     FOREIGN KEY (`gridID`) REFERENCES `grid` (`gridID`),
120     FOREIGN KEY (`npcID`) REFERENCES `npc` (`npcID`),
121     FOREIGN KEY (`itemID`) REFERENCES `item` (`itemID`),
122     FOREIGN KEY (`username`) REFERENCES `character` (`username`)
123 );
```

1. Creating the tile table which is identified through a composite key of the gridID foreign key and the tile coordinate. A tile may be an NPC, character, or item which are joined through nullable foreign keys.

Player Inventory

```
/* INVENTORY CREATE TABLE */

CREATE TABLE `inventory`(
    `username` VARCHAR(50),
    `itemID` INT,
    `quantity` INT NULL,
    PRIMARY KEY (`username`, `itemID`),
    FOREIGN KEY (`username`) REFERENCES `character` (`username`),
    FOREIGN KEY (`itemID`) REFERENCES `item` (`itemID`)
);
```

1. Creating the inventory table which will be identified through the username foreign key of the table owner and the itemID foreign key of the item stored in the inventory.

References

IBM. (2024). *DROP TABLE Statement*. https://www.ibm.com/docs/en/informix-servers/12.10?topic=statements-drop-table-statement

Ravikiran, A. S. (2022, July 8). *Composite key in SQL: Your ultimate guide to mastery [Updated]*. Simplilearn. https://www.simplilearn.com/tutorials/sql-tutorial/composite-key-in-sql

W3 Schools. (2024). *SQL UNIQUE constraint*. W3Schools. https://www.w3schools.com/sql/sql_unique.asp